Hochschule Neu-Ulm
University of Applied Sciences

Technische Hochschule Ulm

Bachelorarbeit

im Bachelorstudiengang

Wirtschaftsingenieurwesen

an der Hochschule für angewandte Wissenschaften Neu-Ulm und der Technischen Hochschule Ulm

KI-Agenten im Forecasting von Supply Chains

Erstkorrektor: Prof. Dr. Tobias Engel

Zweitkorrektor: Prof. Dr. Gunther May

Verfasser: Michael Weber (Matrikel-Nr.: 327185)

Thema erhalten: 01.03.2026

Arbeit abgegeben: 22.06.2026

Abstract

Die Bedarfsprognose bildet die Grundlage der Planung im Supply Chain Management, gerät mit steigender Komplexität, fehleranfälliger Datenbasis und kaum beherrschbarer Informationsmenge jedoch zunehmend an die Grenzen klassischer, manuell gestützter Ansätze. Generative KI in Form LLM-basierter Multi-Agenten-Systeme eröffnet hier einen Lösungsansatz, wurde bislang aber nicht mit der Bedarfsprognose als zentralem Untersuchungsgegenstand erforscht. Die vorliegende Arbeit entwickelt und evaluiert daher ein prototypisches Artefakt: eine Architektur LLM-basierter KI-Agenten für das Forecasting auf Basis historischer Nachfragedaten. Methodisch folgt sie der gestaltungsorientierten Forschung. Aus einer systematischen Literaturrecherche wird eine Liste technologischer Anforderungen abgeleitet, auf deren Grundlage eine Fünf-Agenten-Architektur in CrewAI konstruiert und an die Datenbank eines Planspiels angebunden wird. Die Evaluation untersucht 50 Reports bei identischem Input entlang dreier Stränge: Variabilitätsanalyse, benchmark-basierter Genauigkeitsbewertung und qualitativer Inhaltsanalyse. Die Outputs streuen stark und unterschreiten den optimal parametrisierten Referenzwert systematisch und enthalten teils faktenwidrige Diagnosen, während die managementorientierte Aufbereitung in ihrer Form zuverlässig gelingt. Der Nutzen der Architektur liegt damit in der Unterstützung des Forecasting-Prozesses, nicht in der autonomen, präzisen Prognose, und untermauert die Notwendigkeit eines Human-in-the-Loop bei wirkungsstarken Prognoseentscheidungen.

Key Words

LLM-based agents, multi-agent architecture, demand forecasting, supply chain management

Inhaltsverzeichnis

Abkürzungsverzeichnis

AI	Artificial Intelligence
API	Application Programming Interface
BPOS	Business Process Operations Simulation Game
FF	Forschungsfrage
KI	Künstliche Intelligenz
LLM	Large Language Model
MAD	Mean Absolute Deviation
MAPE	Mean Absolute Percentage Error
MAS	Multi-Agenten-System
MSE	Mean Squared Error
OECD	Organisation for Economic Co-operation and Development
OWASP	Open Worldwide Application Security Project
RMSE	Root Mean Squared Error
SBA	Syntetos-Boylan-Approximation
SCM	Supply Chain Management
SES	Single Exponential Smoothing

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Die Bedarfsprognose bildet die Grundlage für die Planung und Ausführung von Aktivitäten im Supply Chain Management (SCM) (Chopra & Meindl, 2019, S. 186). Mit der zunehmenden Komplexität von Lieferketten steigt jedoch die Unsicherheit, und mit ihr die Wahrscheinlichkeit, dass der Prognosefehler zunimmt, bis zu dem Punkt, an dem das klassische, prognosebasierte Managementmodell grundsätzlich infrage gestellt wird (Christopher, 2023, S. 174). Mit wachsender Netzwerkkomplexität verlieren konventionelle Forecasting-Werkzeuge an Wirksamkeit, weil die Reaktionen des Gesamtsystems auf Störungen kaum noch vorhersehbar sind (Christopher, 2023, S. 175). Verschärft wird dies durch die Datenlage. In heutigen End-to-End-Lieferketten fließen enorme Datenmengen in alle Richtungen, doch sind diese nicht immer korrekt und anfällig für Fehlinterpretationen. Werden Nachfrage- und Angebotsinformationen über mehrere Stufen gefiltert und verändert, gehen verzerrte Daten in Planung und Prognose ein, was deren Genauigkeit senkt (Christopher, 2023, S. 179). Zugleich sind Wertschöpfungsnetze vielfach so komplex, dass die Zahl der planungsrelevanten Informationen für einzelne Menschen kaum noch zu überschauen ist (Arndt, 2021, S. 199), obwohl eine optimale Planung die Verfügbarkeit von Informationen über künftige Bedarfe und vorgelagerte Stufen voraussetzt (Arndt, 2021, S. 191). Damit verdichtet sich das Problem in Lieferketten: Steigende Komplexität und Unsicherheit, eine wachsende und zugleich fehleranfällige Datenbasis sowie eine für Einzelne kaum noch beherrschbare Informationsmenge machen die Bedarfsprognose anspruchsvoller, während klassische, manuell gestützte Ansätze an ihre Wirksamkeits- und Skalierungsgrenzen stoßen. Einen Lösungsansatz eröffnet die generative Künstliche Intelligenz (KI). Sie ermöglicht es, auf eine Eingabe hin automatisch Inhalte zu erzeugen und menschenähnliche Ausgaben zu generieren. Der zentrale Zugang erfolgt über ein auf großen Datenmengen trainiertes Large Language Model (LLM), das Aufgaben wie die Zuordnung von Daten oder deren Zusammenfassung bewältigt (Taulli & Deshmukh, 2025, S. 27). Den architektonischen Kern moderner LLMs bildet das Transformer-Modell, das Daten parallel statt sequenziell verarbeitet und dadurch frühere Ansätze übertrifft (Taulli & Deshmukh, 2025, S. 28–29). Auf dieser Grundlage lassen sich generative KI-Agenten bilden, die Werkzeuge nutzen, Abläufe planen und, in einer Multi-Agenten-Kollaboration arbeitsteilig zusammenwirkend, was einzelnen Agenten häufig überlegen ist, komplexe Aufgaben bearbeiten, wobei menschliche Aufsicht unverzichtbar bleibt (Taulli & Deshmukh, 2025, S. 5–12). Diese Ansätze finden je für sich bereits Anwendung im Forecasting von Lieferketten: Generative KI ist im Supply Chain Management gerade dort das am stärksten bearbeitete Einsatzfeld (Panja, Zheng & Glock, 2026, S. 8), und für autonome Agentensysteme besteht eine ausgereifte Architekturtradition (Xu, Mak & Brintrup, 2021, S. 1). Erst jüngst verbinden LLM-basierte Multi-Agenten-Systeme beides, richten sich aber auf andere Gegenstände wie die Risikobewertung (Quan, Liu, Benaben & Montreuil, 2026) oder das Bestandsmanagement (Jannelli, Schoepf, Bickel, Netland & Brintrup, 2026, S. 14). Ein LLM-basiertes Multi-Agenten-System mit der Bedarfsprognose als zentralem Untersuchungsgegenstand liegt bislang nicht vor.

Genau hier setzt die vorliegende Arbeit an. Ziel ist die Entwicklung und explorative Evaluation eines ersten prototypischen Artefakts in Form einer Architektur LLM-basierter KI-Agenten für das Forecasting im Supply Chain Management. Der Prototyp setzt am Kern des dargelegten Problems an, der Bedarfsprognose auf Basis historischer Nachfragedaten, und erhebt ausdrücklich nicht den Anspruch, das gesamte dargestellte Problemfeld vollständig zu lösen. Vielmehr soll er eine erste, belastbare Einschätzung ermöglichen, inwieweit ein solcher Ansatz das Forecasting im Supply Chain Management unterstützen kann. Das Vorgehen und die dazugehörigen Forschungsfragen werden in Kapitel 2 dargelegt.

2. Forschungsdesign

2.1 Forschungsdesign inkl. Forschungsfragen

FF	Forschungsfrage	Methodik	Erwartetes Ergebnis	Zeit
1	Was sind die technologischen Anforderungen an KI-Agenten mit Fokus auf dem SCM-Bereich?	Literaturrecherche (Webster und Watson, 2002)	Verständnis, wie KI-Agenten gebaut werden sowie daraus abgeleitet das Qualifikationsprofil für Entwickler	11-12/25
2	Wie sieht ein Konzept / Architektur für einen KI-Agenten mit Fokus auf das Forecasting im SCM aus?	Design Science (Hevner)	Architektur/Konzept für die Entwicklung von KI-Agenten.	1-2/26
3	Inwieweit können KI-Agenten bei der Optimierung des Forecastings im SCM behilflich sein?	Gemischte Form der Bewertung (s. Kapitel 2.2.)	Bewertung und Evaluation der entwickelten KI-Agenten-Architektur anhand des Simulationsspiels (BPOS)	3-6/26

Tabelle 1 - Forschungsdesign

Die übergreifende Forschungsmethode ist Design Science nach Hevner, March, Park und Ram (2004). Grundlage bildet die Literaturrecherche nach Webster und Watson (2002) sowie die anwendungsorientierte Entwicklung von KI-Agenten und deren Evaluation mit Hilfe des BPOS-Games.

2.2 Methodisches Vorgehen

Die Literaturrecherche dieser Arbeit verfolgt einen zweistufigen Ansatz. Für die theoretischen Grundlagen wurden Standardbegriffe, u. a. wie Forecasting, KI und KI-Agenten explorativ über fachlich einschlägige Datenbanken (u. a. Springer, Ebscohost, ProQuest, AIS Electronic Library) sowie mittels ergänzender Hilfsmittel (Google Scholar, Claude, ChatGPT) recherchiert, um etablierte Konzepte und Definitionen zu erschließen. Für die Aufarbeitung des Forschungsstands zu KI-Agenten im Supply Chain Management und der ersten Forschungsfrage wurde demgegenüber eine systematische Literaturrecherche nach Webster und Watson (2002) durchgeführt. Über die HNU-Datenbanken ProQuest, EBSCOhost, die AIS Electronic Library und IEEE Xplore wurde anhand definierter Suchstrings in den Feldern Titel, Abstract und Schlagworte gesucht. Die ermittelten Treffer wurden nach Relevanz sortiert und anschließend durch eine Rückwärts- und Vorwärtssuche zu einer themenzentrierten Literaturbasis erweitert (Webster & Watson, 2002, S. xv–xvi). Die Selektion erfolgte anhand vorab definierter Relevanzkriterien: Eingeschlossen wurden begutachtete Beiträge mit Bezug zu KI-Agenten im Forecasting- bzw. Supply-Chain-Kontext, die bei Anwendungszeitschriften mindestens dem VHB-JOURQUAL-Rang B entsprechen, im Volltext zugänglich sind und ab dem Jahr 2016 veröffentlicht wurden. Ausgeschlossen wurden nicht begutachtete, thematisch

unpassende oder nicht verfügbare Quellen. Datenbanken, Suchfelder und Suchbegriffe für beide Literaturrecherchen sind in den nachfolgenden Tabellen dokumentiert.

Datenbank	Ebscohost	AIS eLibrary	ProQuest	IEEE Xplore
Filter	Peer-reviewed + Full-text	Peer-reviewed	Peer-reviewed + Full-text	-
Suchstring	(„large language model*" OR „LLM“ OR „generative AI" OR „GenAI“ OR „foundation model*" OR „GPT“) AND („agent*“ OR „multi-agent" OR „agentic“) AND („forecast*“ OR „demand planning" OR „demand prediction") AND („supply chain*" OR „SCM“)			
Gesucht in	Titel des Dokuments (TI), Abstract (AB), Schlagwort (SU)	TI, AB, SU	TI, AB, SU	TI, AB
Gefunden/Relevant	0|0	0|0	1|0	1|1
Suchstring	(„large language model*" OR „LLM“ OR „generative AI" OR „generative artificial intelligence" OR „GenAI“ OR „foundation model*" OR „GPT“) AND („agent*“ OR „multi-agent" OR „agentic“) AND („supply chain*" OR „SCM“)			
Gesucht in	TI, AB, SU	TI, AB, SU	TI, AB, SU	TI, AB
Gefunden/Relevant	4|2	1|0	9|0	19|1
Suchstring	(„large language model*" OR „LLM“ OR „generative AI" OR „GenAI“ OR „foundation model*" OR „GPT“) AND („forecast*“ OR „demand planning" OR „demand prediction") AND („supply chain*" OR „SCM“)			
Gefunden/Relevant	6|4	0|0	4|0	20|4
Rückwärtssuche	24 gefunden | 5 relevant			
Vorwärtssuche	20 gefunden | 1 relevant			

Tabelle 2 – Darstellung Literaturrecherche zum Stand der Forschung
Quelle: Webster und Watson (2002): Analyzing the past to prepare for the future: Writing a literature review.

Datenbank	Ebscohost	AIS eLibrary	ProQuest	IEEE Xplore
Filter	Peer-reviewed + Full-text	Peer-reviewed	Peer-reviewed + Full-text	-
Suchlauf	Kern FF1: Allgemeine technologische Anforderungen/Architektur von KI-Agenten allgemein.		Aufgrund sehr hoher Trefferanzahl, wurde Suchstring nicht wie in den Datenbanken EBSCOhost und AIS e-Library benutzt, sondern eine Begrenzung auf LLM x	

Datenbank	Ebscohost	AIS eLibrary	ProQuest	IEEE Xplore
			Agent x Architektur x Forecasting-Bezug angewandt. Dies deckt sich mit dem Artefakt dieser Arbeit.	
Suchstring	(„LLM-based agent*" OR „AI agent*" OR „multi-agent system*" OR „autonomous agent*" OR "language model agent*") AND („architecture“ OR „framework“ OR „requirement*“ OR "design pattern*" OR „develop*“)		(„large language model*“ OR LLM OR „generative AI" OR „generative artificial intelligence" OR „GPT“ OR „foundation model*" OR „agentic AI") AND („agent*“ OR „multi-agent" OR „autonomous“) AND („architecture“ OR „framework“ OR „requirement*“ OR "design pattern*" OR „develop*“) AND („forecast*“ OR „demand planning" OR „demand prediction")	
Gesucht in	TI, AB, SU	TI, AB, SU	TI, AB, SU	TI, AB
Treffer/Relevant	86|0	88|5	313 (15) *|1	24|0
Rückwärtssuche	21 Treffer | 5 relevant			
Vorwärtssuche	4 Treffer | 1 relevant			
*Treffer in Journals mit mindestens VHB-JOURQUAL-Rang B				

Tabelle 3 - Darstellung Literaturrecherche zur FF1
Quelle: Webster und Watson (2002): Analyzing the past to prepare for the future: Writing a literature review.

Übergeordnet erfolgt die Entwicklung des Artefakts einer KI-Agenten Architektur dem Schema der gestaltungsorientierten Forschung (Design Science Research) nach Hevner et al. (2004). Den Bezugsrahmen bildet das Drei-Zyklen-Modell von Hevner (2007), bestehend aus Relevanz-, Rigor- und Design-Cycle (Hevner, 2007, S. 88). Im Zentrum steht die Konstruktion eines Artefakts. Damit ist die Anforderung erfüllt, dass gestaltungsorientierte Forschung ein zweckmäßiges Artefakt in Form eines Konstrukts, Modells, einer Methode oder einer Instanziierung hervorbringen muss (Hevner, March, Park & Ram, 2004, S. 83). Das Artefakt ist, eine in CrewAI, realisierte Multi-Agenten-Architektur, bestehend aus generativen KI-Agenten zum Forecasting im SCM. Der wissenschaftliche Beitrag liegt primär in der Entwicklung dieses Artefakts. Das Artefakt ist dabei Bestandteil des Design Cycle, der zwischen dem Entwickeln und dem Evaluieren des Artefakts iteriert (Hevner, 2007, S. 88) und über wiederholte Durchläufe aus Konstruktion, Evaluation und Rückkopplung den Entwurf verfeinert (Hevner et al., 2004, S. 80). Die nachfolgende Abbildung 1 zeigt das auf diese Arbeit angewendete Drei-Zyklen-Modell.

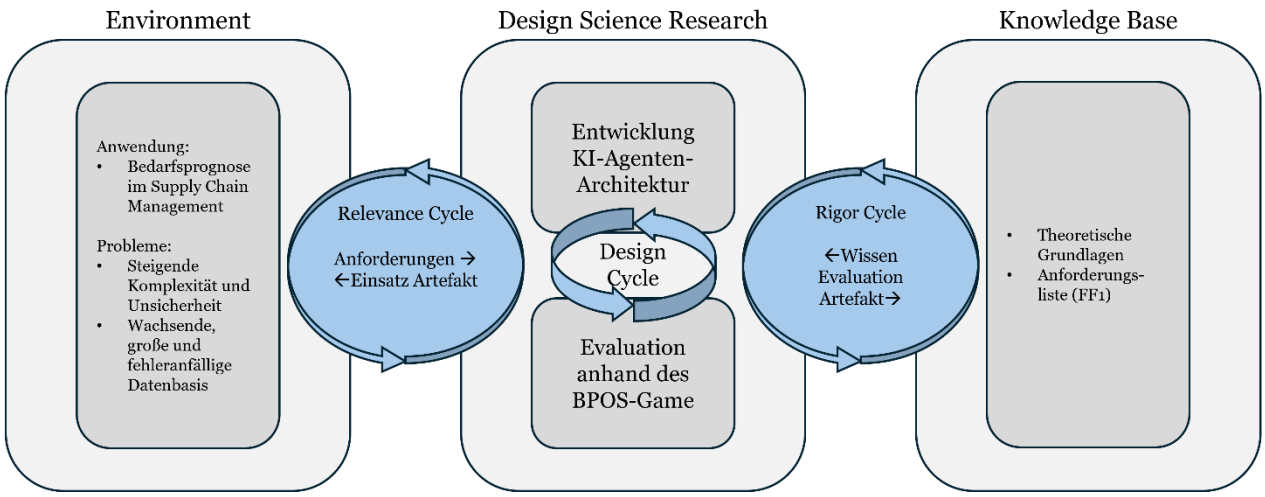


Abbildung 1 – Angewendetes Drei-Zyklen-Modell des Design Science Research
Quelle: Hevner (2007): A three cycle view of design science research.

Abschließend wird die KI-Agenten-Architektur daraufhin evaluiert, inwieweit KI-Agenten das Forecasting von Supply Chains unterstützen können. Die Evaluation ist ein Zusammenspiel aus drei unterschiedlichen Bewertungen, die nachfolgend erklärt werden. Dabei werden alle drei aufeinander bezogenen Stränge auf denselben Textkorpus von 50 Reports der KI-Agenten-Architektur angewendet.

Strang 1: Deskriptive quantitative Inhaltsanalyse. Dieser Strang erhebt formale und inhaltliche Merkmale der Agenten-Outputs regelgeleitet und stellt sie deskriptivstatistisch dar (Döring, 2016, S. 545). Über alle 50 Reports wird anhand eines aus dem Material abgeleiteten Kategoriensystems ausgezählt, ob und wie häufig Fehler auftreten, in welchem Wertebereich die Prognosen liegen und welches Prognosemodell verwendet wurde. Da die Kategorien induktiv aus den Reports gebildet werden, liegt ein datengeneriertes Vorgehen vor, das innerhalb einer quantitativen Inhaltsanalyse ausdrücklich zulässig ist (Döring, 2016, S. 533). Die Inhaltsanalyse dient hier ausschließlich der „systematischen, intersubjektiv nachvollziehbaren Beschreibung inhaltlicher und formaler Merkmale von Mitteilungen" (Früh, 2017, S. 29, zitiert nach Döring, 2016, S. 546). Eine Bewertung oder Inferenz erfolgt nicht, da das Beschreiben und Vergleichen das Ergebnis bildet und das Bewerten lediglich eine zusätzliche, nicht notwendige Option darstellt (Döring, 2016, S. 546). Im Ergebnis entstehen Häufigkeitsdarstellungen, etwa der Anteil der Reports, in denen ein Merkmal auftritt (Döring, 2016, S. 547). Erkenntnisziel ist die Variabilität der Agenten-Outputs bei identischem Input sowie beobachtbare Fehler innerhalb der Arbeitsweise der KI-Agenten.

Strang 2: Benchmark-basierte Genauigkeitsbewertung. Der zweite Strang stellt die Agenten-Outputs einem unabhängig berechneten Referenzmaßstab gegenüber. Auf Basis derselben Daten wurde in einem Tabellenkalkulationsprogramm die fehlerärmste Prognose (geringster Prognosefehler) bestimmt und den Prognosen der 50 Reports gegenübergestellt. Verglichen werden die Modellwahl-Trefferquote, also der Anteil der Reports, in denen das laut Referenz fehlerärmste Modell gewählt wurde, sowie die Abweichung der Agenten-Prognosen vom Referenzwert. Methodisch knüpft dieser Strang an die quantitative Inhaltsanalyse an: Die im ersten Schritt kodierten Merkmale (Modellwahl, Prognosewerte) stellen Messwerte dar, die anschließend mit deskriptiv- und inferenzstatistischen Verfahren ausgewertet werden (Döring, 2016, S. 545). Im Sinne der gestaltungsorientierten Forschung handelt es sich um eine analytische Evaluation, die das Artefakt an einer optimalen Lösung misst (Hevner et al., 2004, S. 85–86). Auch in der Forecasting-Praxis dient Benchmarking als Grundlage zur Bewertung neuer Modelle, indem diese mit etablierten Standardmodellen verglichen werden (Petropoulos, Apiletti, Assimakopoulos, Babai, Barrow, Taieb et al., 2022, S. 758).

Strang 3: Qualitative Inhaltsanalyse nach Mayring (2014). Der dritte Strang beurteilt die inhaltliche Qualität der Bewertungen des management_advisor über alle 50 Reports. Methodisch kommt eine strukturierende Inhaltsanalyse nach Mayring (2014) in ihrer skalierenden Variante zum Einsatz, bei der die Erhebungseinheiten je Dimension auf einer Ordinalskala eingeordnet werden (Mayring, 2014, S. 95). Das Kategoriensystem wurde in einer deduktiv-induktiven Mischform entwickelt: Die acht Strukturierungsdimensionen leiten sich deduktiv aus der Soll-Struktur des management_report ab (siehe. Konfiguration evaluate_management): Nutzen der Prognose, Handlungsempfehlung, Eignung des Verfahrens, Risiken sowie Belastbarkeit je Produkt (prod13–16) und kritische Methodenbewertung, Risikoanalyse sowie Gesamturteil produktübergreifend. Die Ausprägungen je Dimension (hoch, mittel, niedrig, fehlend) und die zugehörigen Kodierregeln wurden demgegenüber induktiv aus dem Material der 50 Reports gewonnen. Dem Ablaufmodell folgend wurde das Material durchgearbeitet, das Kategoriensystem nach einem ersten Materialdurchgang überarbeitet und das Material anschließend vollständig final kodiert (Mayring, 2014, S. 80, S. 98). Das resultierende Kategoriensystem besteht aus Kategoriendefinitionen, Ankerbeispielen und Kodierregeln (Mayring, 2014, S. 101). Die Kodierung erfolgte rechnergestützt mit der Software ATLAS.ti. Die vergebenen Codes wurden anschließend in ein Tabellenkalkulationsprogramm exportiert und zur besseren Übersicht mit Häufigkeiten versehen. Auf dieser Grundlage werden im Folgenden die Häufigkeiten der zugeordneten Kategorien über alle Erhebungseinheiten ausgewertet (Mayring, 2014, S. 98), womit der qualitative Strang den quantitativen Auswertungsschritt integriert.

3. Theoretische Grundlagen

3.1 Grundlagen Supply Chain Management

Eine Supply Chain (Lieferkette) umfasst alle Parteien, die direkt oder indirekt an der Erfüllung einer Kundenanfrage beteiligt sind. Dazu gehören nicht nur der Hersteller und die Lieferanten, sondern auch Transporteure, Lagerhäuser, Einzelhändler und sogar die Kunden selbst (Chopra & Meindl, 2019, S. 15). Dabei durchziehen drei zentrale Flüsse die Supply Chain: Informationen, Produkte und Geldmittel (Chopra & Meindl, 2019, S. 16).

Der Begriff „Supply Chain Management“ (SCM) ist 1982 von Oliver und Webber erstmals anhand von vier Merkmalen beschrieben worden (Oliver und Webber, 1982, zitiert nach Klaus & Müller, 2012). Der Anwendungsbereich wurde dabei als „Supply-chain management umfasst den Warenfluss vom Lieferanten über die Fertigungs- und Vertriebsketten bis hin zum Endverbraucher“ beschrieben (Oliver und Webber, 1982, zitiert nach Klaus & Müller, 2012). Nach Christopher ist die heutige Definition von SCM „Die Steuerung der Beziehungen zu Lieferanten und Kunden in vor- und nachgelagerten Bereichen, um einen höheren Kundennutzen bei geringeren Kosten für die gesamte Lieferkette zu erzielen“ (Christopher, 2023, S. 3). SCM ist somit ein umfassendes Konzept zur Steuerung der gesamten Wertschöpfungskette (Christopher, 2023, S. 3). Aus dieser unternehmensübergreifenden Ausrichtung folgt ein verändertes Wettbewerbsverständnis. Nicht mehr einzelne Unternehmen, sondern ganze Wertschöpfungsketten konkurrieren miteinander. Ein zentraler Gedanke des SCM besteht entsprechend darin, durch die Reduktion von Schnittstellen und die Verbesserung der Flüsse eine tatsächliche Verbesserung der gesamten Supply Chain zu erreichen, statt Kosten lediglich auf Lieferanten abzuwälzen (Arndt, 2021, S. 46). Diese Form der Verbesserung beinhaltet die Maximierung des Nettonutzens der Supply Chain. Der Nettonutzen ergibt sich dabei aus der Differenz zwischen dem, was das Endprodukt für den Kunden wert ist und den Kosten der gesamten Supply Chain (Chopra & Meindl, 2019, S. 18–19). Das Planen innerhalb der Supply Chain vollzieht sich auf drei unterschiedlichen zeitlichen Ebenen. Die strategische Ebene legt über mehrere Jahre die Konfiguration der Supply Chain fest, wie z. B. Standorte, Kapazitäten, Eigen- oder Fremderstellung sowie die einzusetzenden Informationssysteme. Die taktische Planungsebene umfasst einen Horizont von einem Quartal bis zu einem Jahr. Sie operiert innerhalb der in der strategischen Ebene

festgelegten Konfiguration und hat das Ziel, den Nettonutzen der Supply Chain zu maximieren. Der Start erfolgt typischerweise mit einer Prognose der Nachfrage sowie weiterer Größen wie Kosten und Preise für den kommenden Zeitraum. Die operative Ebene schließlich bezieht sich auf den wöchentlichen bis täglichen Horizont konkreter Kundenaufträge, in dem Konfiguration und Planungsrichtlinien als gegeben gelten. Jede Ebene definiert dabei die Rahmenbedingungen für die jeweils nachgelagerte (Chopra & Meindl, 2019, S. 20–22). Abläufe und die Flüsse, die in der Supply Chain stattfinden, lassen sich in zwei Sichtweisen unterscheiden. Die Zyklus-Sicht gliedert die Prozesse in aufeinanderfolgende Zyklen an den Schnittstellen je zweier Stufen: Kundenauftrags-, Wiederbeschaffungs-, Fertigungs- und Beschaffungszyklus. Die Push-/Pull-Sicht ordnet die Prozesse demgegenüber danach, ob sie als Reaktion auf einen konkreten Kundenauftrag oder in Erwartung künftiger Aufträge ausgeführt werden. Alle Prozesse einer Supply Chain fallen dabei in zwei Kategorien. Pull-Prozesse sind reaktiv und werden durch einen vorliegenden Auftrag ausgelöst. Push-Prozesse hingegen sind spekulativ und werden in Antizipation der Nachfrage auf Basis einer Prognose angestoßen (Chopra & Meindl, 2019, S. 22–24). Bedarfsprognosen sind dabei die Basis in beiden Kategorien. Denn in Push-Prozessen muss die Nachfrage direkt antizipiert werden, um die Produktion oder den Transport direkt darauf auszurichten. Während in Pull-Prozessen nicht direkt der tatsächliche Bedarf geplant werden muss, so muss dennoch prognostiziert werden, was die Nachfrage sein wird, denn verfügbare Kapazitäten und Lager müssen darauf geplant werden (Chopra & Meindl, 2019, S. 186).

3.2 Grundlagen Forecasting

Die Erstellung von Prognosen über die Nachfrage beschreibt das „Forecasting“. Das Forecasting geht davon aus, dass vergangenes Wissen genutzt werden kann, um Prognosen über die Zukunft zu treffen. Vor allem in Bezug auf Zeitreihen beruht dies auf der Annahme, dass erkennbare Muster in historischen Daten eine Vorhersage zukünftiger Werte ermöglichen. Eine Prognose beinhaltet dabei verschiedenste Optionen, wie ein Erwartungswert (Punktprognose), Prognoseintervall, ein Perzentil und eine gesamte Prognoseverteilung (Petropoulos et al., 2022, S. 710–711). Zu beachten hierbei ist, dass Prognosen immer falsch sind und deshalb sowohl den erwarteten Wert der Prognose als auch eine Messung des Prognosefehlers beinhalten soll (Chopra & Meindl, 2019, S. 187). Der Aufbau einer Prognose setzt sich zusammen, indem die beobachtbare Nachfrage in zwei Komponenten aufgeteilt wird: Beobachtete Nachfrage = systematische Komponente + zufällige Komponente. Die systematische Komponente misst dabei den erwarteten Nachfragewert, bestehend aus Niveau, Trend und Saisonalität. Das Niveau beschreibt die aktuelle desaisonalisierte Nachfrage (Chopra & Meindl, 2019, S. 189). Der Trend beschreibt die glatte, zugrundeliegende mittlere Entwicklung einer Zeitreihe über einen längeren Zeitraum, während die Saisonalität ein sich regelmäßig wiederholendes Muster innerhalb eines festen Zeitraums beschreibt (Petropoulos et al., 2022, S. 712). Die zufällige Komponente ist der Teil der Nachfrage, der vom systematischen Teil abweicht. Die zufällige Komponente lässt sich nicht prognostizieren, aber abschätzbar sind ihre erwartete Größe und Variabilität, was zugleich ein Maß für Prognosefehler liefert. Daraus folgt das Ziel jeder Prognose: das Herausfiltern der zufälligen Komponente und das Schätzen der systematischen Komponente. Der Prognosefehler misst dabei die Differenz zwischen Prognose und tatsächlicher Nachfrage (Chopra & Meindl, 2019, S. 189).

3.2.1 Forecasting-Methoden

Innerhalb des Forecasting gibt es eine Vielzahl an Methoden. Diese lassen sich in vier Hauptgruppen einteilen: Qualitative Verfahren, Zeitreihenverfahren, Kausale Verfahren und Simulationsverfahren. Qualitative Methoden sind subjektiv und stützen sich auf das menschliche Urteilsvermögen. Diese Methoden sind am besten geeignet, wenn nur wenig historische Daten verfügbar sind oder Experten über Marktwissen verfügen, das Prognosen beeinflussen kann. Zeitreihenverfahren leiten die künftige Nachfrage aus der vergangenen ab und setzen voraus, dass sich das Nachfragemuster über die

Zeit nur wenig ändert. Sie sind technisch am unkompliziertesten und dienen häufig als erster Anhaltspunkt. Kausale Verfahren gehen davon aus, dass die Nachfrage eng mit äußeren Einflussgrößen zusammenhängt, etwa der wirtschaftlichen Lage oder dem Preisniveau. Über diesen Zusammenhang lässt sich abschätzen, wie sich beispielsweise eine Preisaktion auf die Nachfrage auswirkt. Simulationsverfahren bilden das Kaufverhalten der Kunden nach und verbinden dabei Zeitreihen- und kausale Ansätze. So lassen sich Szenarien durchspielen, wie die Folgen einer Preisaktion oder eines neuen Wettbewerbers in der Nähe. Fluggesellschaften nutzen dies, um die Nachfrage nach höherpreisigen Tickets vorherzusagen (Chopra & Meindl, 2019, S. 189). Im Folgenden werden diejenigen Forecasting-Methoden erklärt, welche später in dieser Arbeit Anwendung finden.

(1) Naive Prognose

Die Naive Prognose funktioniert wie folgt: alle Prognosen werden einfach auf den zuletzt beobachteten Wert gesetzt:

$$y_{t+h} = y_t$$

Dabei ist y_t der letzte beobachtete Bedarf und y_{t+h} der prognostizierte Bedarf für h Perioden in der Zukunft (Hyndman & Athanasopoulos, 2021, Abschnitt 5.2).

Die Exponentielle Glättung ist ein weit verbreitetes Verfahren im Forecasting, das auf einem gewichteten Durchschnitt vergangener Beobachtungen basiert. Das Grundprinzip besteht darin, dass die Gewichtung für vergangene Daten exponentiell abnimmt, je weiter sie in der Vergangenheit liegen. Trotz vieler Fortschritte im Forecasting gilt sie als guter Maßstab, den man im Auge behalten sollte. Die Exponentielle Glättung lässt sich in drei Formen unterscheiden (Petropoulos et al., 2022, S. 715).

(2) Einfache Exponentielle Glättung

Eine davon ist die Einfache Exponentielle Glättung, welche sich mit folgender Formel berechnen lässt:

$$L_t = \alpha \times D_t + (1 - \alpha) L_{t-1}$$

L_t = geschätztes Niveau am Ende der aktuellen Periode t

L_{t-1} = geschätztes Niveau der vorherigen Periode t-1

D_t = tatsächlicher Bedarf, beobachtet in der aktuellen Periode t

α = Glättungsfaktor Alpha, liegt zwischen 0 und 1

F_{t+1} = Prognose für die nächste Periode, wobei $F_{t+1} = L_t$

Die Einfache Exponentielle Glättung berechnet das geschätzte Niveau am Ende einer Periode, indem zuerst der Glättungsfaktor α eine Gewichtung zwischen dem aktuellsten beobachteten Bedarf D_t und dem vorherigen geschätzten Niveau L_{t-1} vornimmt. Anschließend bilden diese gewichteten Teile zusammenaddiert das geschätzte Niveau am Ende der Periode L_t und somit auch für die Prognose der nächsten Periode F_{t+1} . Um die Einfache Exponentielle Glättung zu benutzen, benötigt man das vorherig geschätzte Niveau L_{t-1}, welches aus dem Durchschnitt der vergangenen tatsächlichen Bedarfe entnommen werden kann. Außerdem benötigt man einen Wert für den Glättungsfaktor α (Die Wahl aller Glättungsfaktoren wird nach (4) dargelegt). Dieser liegt zwischen 0 und 1 und steuert die Reagibilität der Prognose. Bei einem niedrigen α -Wert ist die Prognose stabiler und lässt sich weniger von plötzlich starken Nachfrageänderungen beeinflussen, während bei einem höheren α -Wert die Prognose schneller auf aktuelle Änderungen reagiert. In der Praxis wird oft ein α nicht größer als 0,2 empfohlen, sofern das zugrunde liegende Nachfragemuster stabil ist (Chopra & Meindl, 2019, S. 198–199).

(3) Trendkorrigierte Exponentielle Glättung (Holts Modell)

Eine weitere Form von Exponentielle Glättung ist die Trendkorrigierte Exponentielle Glättung, auch bekannt als „Holts Modell". Es erweitert die Einfache Exponentielle Glättung ausschließlich um die Komponente „Trend". Dieses Modell schätzt dabei zwei Komponenten kontinuierlich neu, sobald eine neue Beobachtung eintrifft: das aktuelle Niveau und den Trend. Die Prognose erfolgt anschließend durch Addition dieser beiden Faktoren.

Niveau und Trend werden mittels folgender Formeln geschätzt:

$$L_t = \alpha \times D_t + (1 - \alpha)(L_{t-1} + T_{t-1})$$

$$T_t = \beta \times (L_t - L_{t-1}) + (1 - \beta)T_{t-1}$$

L_t = geschätztes Niveau am Ende der aktuellen Periode t

L_{t-1} = geschätztes Niveau der vorherigen Periode t-1

D_t = tatsächlicher Bedarf, beobachtet in der aktuellen Periode t

T_t = geschätzter Trend am Ende der aktuellen Periode t

T_{t-1} = geschätzter Trend der vorherigen Periode t-1

α = Glättungsfaktor Alpha, liegt zwischen 0 und 1

β = Glättungsfaktor Beta, liegt zwischen 0 und 1

F_{t+1} = Prognose für die nächste Periode, wobei $F_{t+1} = L_t + T_t$

Das geschätzte Niveau L_t ist ein gewichteter Durchschnitt aus dem aktuellsten beobachteten Bedarf D_t und der vorherigen Prognose, bestehend aus geschätztem Niveau der vorherigen Periode L_{t-1} und geschätztem Trend der vorherigen Periode T_{t-1}. Die Gewichtung erfolgt durch den Glättungsfaktor α. Der geschätzte Trend ist ein gewichteter Durchschnitt aus der Veränderung der aktuellen und vorherigen Niveauschätzung $(L_t - L_{t-1})$ und dem geschätzten Trend der vorherigen Periode T_{t-1}. Hierbei erfolgt die Gewichtung mittels Glättungsfaktor β. Um das Modell zu starten, müssen die Anfangswerte L_0 und T_0, also Startwerte für das geschätzte Niveau der vorherigen Periode L_{t-1} und dem geschätzten Trend der vorherigen Periode T_{t-1} festgelegt werden. Dies erfolgt in der Praxis meist durch lineare Regression der ersten verfügbaren historischen Datenpunkte, wobei der Achsenabschnitt als L_0 und die Steigung als T_0 dient. Die Prognose für die nächste Periode F_{t+1} ist die Summe der Komponenten: aktuell geschätztes Niveau L_t und aktuell geschätzter Trend T_t (Chopra & Meindl, 2019, S. 199–201).

(4) Trend- und Saisonalitätkorrigierte Form der Exponentiellen Glättung (Winters-Modell)

Abschließend gibt es noch die Trend- und Saisonalitätkorrigierte Form der Exponentiellen Glättung, auch bekannt als „Winters-Modell". Gegenüber der Trendkorrigierten Exponentiellen Glättung fließt hierbei zusätzlich die Komponente „Saisonalität" in die Prognose ein. Nach Schätzung der Komponenten Niveau, Trend und Saisonalität wird die Prognose durch Kombination dieser Komponenten erstellt.

Das geschätzte Niveau L_t wird wie folgt berechnet:

$$L_t = \alpha \times (D_t/S_t) + (1 - \alpha)(L_{t-1} + T_{t-1})$$

S_t = geschätzte Saisonalität am Ende der aktuellen Periode t

Die Berechnung des geschätzten Niveaus L_t erfolgt durch einen gewichteten Durchschnitt aus dem saisonbereinigten aktuellsten beobachteten Bedarf (D_t/S_t) und der vorherigen Prognose, bestehend aus geschätztem Niveau der vorherigen Periode L_{t-1} und geschätztem Trend der vorherigen Periode

T_{t-1}. Die Gewichtung erfolgt durch den Glättungsfaktor α. Der geschätzte Trend T_t wird gleich wie in der Trendkorrigierten Exponentiellen Glättung berechnet:

$$T_t = \beta \times (L_t - L_{t-1}) + (1 - \beta)T_{t-1}$$

Hinzu kommt die Berechnung der geschätzten Saisonalität S_{t+p} :

$$S_{t+p} = \gamma \times \left(\frac{D_t}{L_t}\right) + (1 - \gamma)S_t$$

S_t = geschätzte Saisonalität für die aktuelle Periode tS_{t+p} = geschätzte Saisonalität, am Ende der aktuellen Periode t, für den nächsten saisonalen Zyklus, der p Perioden entfernt ist

p = Anzahl an Perioden, um den nächsten saisonalen Zyklus zu erreichen

γ = Glättungsfaktor Gamma

Die geschätzte Saisonalität für den nächsten saisonalen Zyklus S_{t+p} ergibt sich aus einem gewichteten Durchschnitt aus dem Verhältnis von tatsächlich beobachtetem Bedarf D_t und geschätztem Niveau L_t und der geschätzten Saisonalität am Ende der aktuellen Periode S_t. Die Gewichtung erfolgt durch den Glättungsfaktor γ. Um mit diesem Modell arbeiten zu können, müssen analog zur Trendkorrigierten Exponentiellen Glättung Startwerte für L_0 und T_0 sowie die Glättungsfaktoren α und β bestimmt werden. Hinzu kommt zusätzlich die Bestimmung für γ (Chopra & Meindl, 2019, S. 201–202). S_t ergibt sich als Verhältnis aus der tatsächlichen Nachfrage zur desaisonalisierten Nachfrage einer Periode t. Die desaisonalisierte Nachfrage besteht aus der Summe von dem Niveau L in Periode 0 und dem Trend multipliziert mit der Anzahl an Perioden t (Chopra & Meindl, 2019, S. 194–195).

Um den optimalen Wert für Glättungsparameter zu finden, minimieren moderne Systeme in der Regel die Summe der quadrierten Vorhersagefehler (Petropoulos et al., 2022, S. 715). Welche Parameter zu schätzen sind, hängt vom gewählten Verfahren ab. Die Einfache Exponentielle Glättung enthält allein den Niveau-Parameter α (Hyndman & Athanasopoulos, 2021, Abschnitt 8.1), Holts-Modell zusätzlich den Trend-Parameter β (Hyndman & Athanasopoulos, 2021, Abschnitt 8.2) und Winters-Modell darüber hinaus den Saison-Parameter γ (Hyndman & Athanasopoulos, 2021, Abschnitt 8.3). Unabhängig vom Verfahren werden diese Glättungsparameter nicht manuell vorgegeben, sondern gemeinsam mit den Startwerten der Komponenten aus den Daten geschätzt. Als Optimierungskriterium dient dabei die Minimierung der Summe der quadrierten Ein-Schritt-Prognosefehler. Die zulässigen Werte sind dabei beschränkt. Traditionell liegen die Parameter zwischen 0 und 1, damit sich die Glättungsgleichungen als gewichtete Mittelwerte interpretieren lassen. In der Zustandsraum-Schreibweise übersetzt sich dies in die Grenzen $0 < \alpha < 1$, $0 < \beta < \alpha$ und $0 < \gamma < 1 - \alpha$. Innerhalb dieser Grenzen wird genau diejenige Kombination aus α, β und γ ausgewählt, die das Optimierungskriterium erfüllt (Hyndman & Athanasopoulos, 2021, Abschnitt 8.6).

(5) Syntetos-Boylan-Approximation

In vielen Fällen lässt sich die Nachfrage in jedem Zeitraum beobachten, allerdings kann die Nachfrage alternativ auch sporadisch auftauchen, wobei in manchen Zeiträumen gar keine Nachfrage besteht. Dies nennt man „Intermittierende Nachfrage“. Intermittierende Nachfrage zeichnet sich durch eine hohe Anzahl von Nullwerten aus, die von zufälligen, positiven Nachfrageereignissen unterbrochen werden. Solche Nachfragemuster treten häufig bei Ersatzteilen in Branchen wie der Automobilindustrie oder Luftfahrt auf. In der Praxis wird die Einfache Exponentielle Glättung oft für intermittierende Nachfrage verwendet. Allerdings berücksichtigt diese Methode die spezielle Nachfrage nicht. Den Maßstab, an dem andere (neue) vorgeschlagene Methoden im Bereich der Prognose für intermittierende Nachfrage gemessen werden, stellt die Syntetos-Boylan-Approximation (SBA) dar (Petropoulos et al., 2022, S. 748).

SBA lässt sich mittels folgender Formel berechnen:

$$\widehat{Y_t} = \left(1 - \frac{\alpha}{2}\right) \frac{\hat{z}_t}{\hat{p}_t}$$

$\widehat{Y_t}$ = geschätzter mittlerer Bedarf pro Periode t, also die eigentliche Prognosegröße

$\hat{z}_t$ = die exponentiell geglättete Schätzung der durchschnittlichen Nachfragehöhe in den Perioden t, in denen überhaupt Nachfrage auftritt.

$\hat{p}_t$ = die exponentiell geglättete Schätzung des durchschnittlichen Abstands zwischen zwei Nachfrageereignissen, also des mittleren Intervalls zwischen Nachfragen

α = Glättungskonstante, die zur Aktualisierung der Intervallschätzung verwendet wird

Die Formel zerlegt intermittierende Nachfrage in zwei Teile: erstens, wie groß eine Nachfrage ist, wenn sie auftritt, nämlich $\hat{z}_t$ und zweitens, wie oft sie auftritt, ausgedrückt über das mittlere Intervall $\hat{p}_t$. Der Quotient $\frac{\hat{z}_t}{\hat{p}_t}$ liefert damit eine Schätzung des durchschnittlichen Bedarfs pro Periode.Die SBA multipliziert diesen Quotienten dann mit $(1 - \frac{\alpha}{2})$ (Syntetos & Boylan, 2005, S. 304). Syntetos und Boylan (2005) zeigen, dass der Quotient nach Crostons Methode verzerrt ist und diese Korrektur zu einer ungefähr unverzerrten Schätzung führt (Syntetos & Boylan, 2005, S. 313).

3.2.2 Prognosefehler

Jede beobachtbare Nachfrage enthält eine zufällige Komponente, die sich einer Prognose entzieht und sich in Form eines Prognosefehlers niederschlägt. Eine geeignete Methode soll die systematische Komponente der Nachfrage abbilden, nicht jedoch die zufällige Komponente erfassen. Die Analyse der Prognosefehler erfüllt dabei zwei Funktionen. Zum einen lässt sich beurteilen, ob die eingesetzte Methode die systematische Komponente zutreffend prognostiziert, denn ein dauerhaft positiver Fehler deutet etwa auf eine systematische Überschätzung hin und sollte korrigiert werden. Zum anderen bildet das Ausmaß des Prognosefehlers die Grundlage für Planungen, wie Kapazitätsplanungen, da sämtliche Notfallpläne auf das erwartete Fehlerausmaß abgestimmt sein müssen (Chopra & Meindl, 2019).

Der Prognosefehler der Periode t ist definiert als Differenz zwischen Prognose F_t und tatsächlicher Nachfrage D_t:

$$E_t = F_t - D_t$$

Der Fehler sollte dabei mindestens für jenen Horizont geschätzt werden, der der für eine Reaktion erforderlichen Vorlaufzeit entspricht.

Ein erstes aggregiertes Fehlermaß ist der mittlere quadratische Fehler (mean squared error, MSE):

$$MSE_n = \frac{1}{n} \sum_{t=1}^{n} E_t^2$$

Der MSE schätzt die Varianz des Prognosefehlers und gewichtet große Abweichungen durch die Quadrierung überproportional stark. Er eignet sich daher vor allem, wenn große Fehler mit hohen Kosten verbunden sind, und setzt eine um null symmetrische Fehlerverteilung voraus.

Als Alternative dient die absolute Abweichung $A_t = |E_t|$ und daraus die mittlere absolute Abweichung (mean absolute deviation, MAD):

$$MAD_n = \frac{1}{n} \sum_{t=1}^{n} A_t$$

Unter der Annahme normalverteilter Fehler lässt sich aus dem MAD die Standardabweichung der zufälligen Komponente abschätzen. Der MAD ist dem MSE überlegen, wenn die Fehlerverteilung nicht symmetrisch ist. Bei ausgeprägter Saisonalität und stark schwankender Nachfrage ist hingegen der mittlere absolute prozentuale Fehler (mean absolute percentage error, MAPE) vorzuziehen, da er die Fehler relativ zur Nachfragehöhe ausdrückt:

$$MAPE_n = \frac{\sum_{t=1}^{n} \left|\frac{E_t}{D_t}\right| \ 100}{n}$$

Darüber hinaus gibt es noch „bias“ und Trackingsignale für verzerrte Prognosen, das wird hier jedoch nicht erklärt (Chopra & Meindl, 2019, S. 203–204).

3.3 Grundlagen Künstliche Intelligenz

Die Künstliche Intelligenz (KI) lässt sich als ein Teilbereich der Informatik einordnen, deren übergeordnetes Ziel der Bau intelligenter Maschinen ist. John McCarthy lieferte 1955 eine erste griffige Definition für „Künstliche Intelligenz“: „Für die vorliegenden Zwecke wird das Problem der künstlichen Intelligenz so definiert, dass eine Maschine sich so verhält, wie man es als intelligent bezeichnen würde, wenn sich ein Mensch entsprechend verhalten würde.“ (Ertel, 2016, S. 1; McCarthy, Minsky, Rochester & Shannon, 2006, S. 9). Die „Organisation for Economic Co-operation and Development“ liefert eine aktuelle Definition von KI-Systemen, die offiziell in der Politik Verwendung findet. Die „Organisation for Economic Co-operation and Development“ (OECD) ist eine internationale Organisation zur Förderung wirtschaftlicher Zusammenarbeit und entwickelt unter anderem internationale Leitlinien und Standards für den verantwortungsvollen Umgang mit Künstlicher Intelligenz. Die OECD hat Ende 2023 ihre Definition eines KI-Systems überarbeitet, um angesichts technologischer Entwicklungen weiterhin relevant und technisch präzise zu bleiben. Somit lautet die aktuelle Definition: „An AI system is a machine-based system that, for explicit or implicit objectives, infers, from the input it receives, how to generate outputs such as predictions, content, recommendations, or decisions that can influence physical or virtual environments. Different AI systems vary in their levels of autonomy and adaptiveness after deployment.“ (Organisation for Economic Co-operation and Development, 2024, S. 4–7). Diese Definition beschreibt KI als maschinenbasierte Systeme, die aus Eingaben eigenständig Ausgaben wie Vorhersagen, Inhalte, Empfehlungen oder Entscheidungen ableiten können, die physische oder virtuelle Umgebungen beeinflussen. Dies spiegelt den heutigen Entwicklungsstand der KI wider, der das Ergebnis einer mehr als 70 Jahre umfassenden Entwicklung ist. Schon 1950 hatte Alan Turing mit dem nach ihm benannten „Turing-Test“ ein operationales Kriterium für Maschinenintelligenz vorgeschlagen, das bis heute als Bezugspunkt dient (Turing, 1950, S. 33). Dabei gelte eine Maschine als intelligent, wenn sie einen menschlichen Prüfer in einem schriftlichen Dialog in mindestens 30 % der Fälle darüber täuschen kann, ob sein Gegenüber Mensch oder Maschine ist (Ertel, 2016, S. 8; Turing, 1950, S. 40). Als eigenständige Wissenschaft etablierte sich die KI auf der Dartmouth-Konferenz 1956, als McCarthy dort den Begriff „Artificial Intelligence“ einführte und Newell und Simon mit dem Logic Theorist das erste symbolverarbeitende Programm vorstellten. Ab diesem Zeitpunkt entwickelte sich die KI über Jahrzehnte hinweg weiter (Ertel, 2016, S. 8; McCarthy et al., 2006, S. 9).

Generative KI bezeichnet einen Teilbereich der künstlichen Intelligenz, der die Erzeugung vielfältiger Inhalte wie Texte, Programmcode, Bilder oder Videos ermöglicht. Angestoßen wird dieser Prozess durch die Eingabe einer Anweisung, eines sogenannten Prompts, woraufhin das System eine menschenähnliche Ausgabe generiert. Der zentrale Zugang erfolgt über ein Large Language Model (LLM), ein auf umfangreichen Datenmengen aus unterschiedlichsten Themenfeldern trainiertes System, das dadurch Aufgaben wie Übersetzung, Zuordnung von Daten zu vordefinierten Gruppen oder Zusammenfassung bewältigt (Taulli & Deshmukh, 2025, S. 27). LLMs sind vortrainiert, das

heißt, sie werden auf großen Datenbeständen aus Korpora wie Wikipedia oder weiteren Internetquellen trainiert (Taulli & Deshmukh, 2025, S. 28). Dabei erlernt das Modell sogenannte Embeddings, das sind dichte Vektorrepräsentationen von Wörtern oder „Tokens“. Ein Token bezeichnet ein Wort, einen Wortbestandteil oder ein einzelnes Zeichen. Wie ein Text in Tokens zerlegt wird, hängt vom jeweiligen LLM ab, und das Token bildet die Einheit, mit der ein Modell Text verarbeitet (Taulli & Deshmukh, 2025, S. 61). Embeddings sind somit Zahlenarrays fester Länge, die semantische Beziehungen zwischen Wörtern erfassen und kontextbezogenes Verstehen und Erzeugen von Text ermöglichen. Im Unterschied zu klassischen KI-Modellen, die auf strukturierten, etikettierten Daten beruhen, erlaubt generative KI ein Training auf unstrukturierten Daten. Verstärkt wird ihre Leistungsfähigkeit durch die „Scaling Laws“, denen zufolge die Leistung mit zunehmender Datenmenge und Modellgröße steigt. Als Nachteile gelten der Datenstichtag, durch den aktuelle Informationen fehlen, sowie die knappe Verfügbarkeit hochwertiger Daten. Dagegen gewinnt die Erzeugung synthetischer Daten, also künstlich durch Algorithmen erzeugter, realitätsnaher Daten, an Bedeutung. Den architektonischen Kern moderner LLMs bildet das „Transformer-Modell“, das die Verarbeitung natürlicher Sprache grundlegend verändert hat. Eingeführt mit dem 2017 erschienenen Aufsatz „Attention Is All You Need", übertrifft es frühere Ansätze wie rekurrente neuronale Netze, da es Daten parallel statt sequenziell verarbeitet (Taulli & Deshmukh, 2025, S. 28–29). Schlüsselkomponente ist dabei der „Attention“-Mechanismus, insbesondere die „Self-Attention“, die die Bedeutung einzelner Tokens innerhalb eines Kontextes gewichtet und so deren Beziehungen erfasst. Unterschieden wird dabei wie folgt: in autoregressive Modelle, die das nächste Token aus den vorausgehenden vorhersagen, in autoencodierende Modelle, die Tokens bidirektional aus dem Kontext ableiten, sowie in Modelle, die beide Ansätze kombinieren (Taulli & Deshmukh, 2025, S. 30).

LLMs lassen sich in drei Typen einteilen: proprietäre Systeme, Open-Source-Modelle sowie kleinere Modelle, die als Small Language Models (SLMs) bezeichnet werden. Proprietäre LLMs sind fortgeschrittene KI-Systeme, die von privaten Organisationen entwickelt, gehalten und kontrolliert werden. Zu den bekanntesten Beispielen zählen die GPT-Modelle von OpenAI, Claude von Anthropic und Gemini von Google. Die Entwicklung der GPT-Modelle von OpenAI veranschaulicht die rasante Leistungssteigerung proprietärer LLMs besonders deutlich. Das erste Modell, GPT-1, wurde 2018 mit 117 Millionen Parametern vorgestellt. Es basierte auf einer Transformer-Architektur, wurde auf einem umfangreichen Korpus von Internettexten trainiert und konnte trotz begrenzter Fähigkeiten bereits kohärente und kontextbezogene Texte erzeugen. In den Folgejahren wuchs die Modellgröße exponentiell: 2019 erschien GPT-2 mit 1,5 Milliarden Parametern, 2020 folgte GPT-3 mit 175 Milliarden Parametern, wodurch ein deutlich breiteres Aufgabenspektrum mit hoher Genauigkeit bewältigt werden konnte. Bei GPT-4 legte OpenAI die genaue Parameterzahl nicht mehr offen. Schätzungen gehen von über einer Billion Parametern aus. Diese zunehmende Intransparenz ist im wettbewerbsintensiven Umfeld mittlerweile verbreitet, da die Anbieter ihre Innovationen schützen wollen (Taulli & Deshmukh, 2025, S. 34–35). Den zweiten Typ bilden Open-Source-LLMs. Sie erlauben es Forschenden, Entwickelnden und Nutzenden, Architektur, Gewichte und teils Trainingsdaten einzusehen, und fördern dadurch Transparenz, gemeinschaftliche Weiterentwicklung sowie eine Demokratisierung des Zugangs. Den dritten Typ stellen Small Language Models dar, die bei geringerem Rechenbedarf eine solide Leistung liefern und sich dadurch für ressourcenbeschränkte Umgebungen sowie für eng umrissene Aufgaben wie Zusammenfassung oder Klassifikation eignen (Taulli & Deshmukh, 2025, S. 38–39).

3.4 Grundlagen KI-Agenten

Ein KI-Agent ist eine digitale Einheit, die über die Fähigkeit verfügt, ihre Umgebung wahrzunehmen, darüber nachzudenken und mit einem hohen Maß an Unabhängigkeit Aktionen auszuführen (Huang, 2025, S. 2). Unterschieden werden kann bei KI-Agenten in zwei unterschiedliche Systeme:

Single-Agenten-Systeme und Multi-Agenten-Systeme (MAS). Beide beruhen auf autonomen Einheiten (agents), die eigenständig Aufgaben bearbeiten. Der grundlegende Unterschied liegt in deren Anzahl und Zusammenspiel. Ein Single-Agenten-System besteht aus einer einzigen autonomen Einheit, die Aufgaben löst, während ein MAS mehrere autonome, miteinander interagierende Einheiten umfasst. Aus dieser strukturellen Differenz ergeben sich mehrere Gegensätze. Hinsichtlich der Autonomie operiert der einzelne Agent unabhängig und ohne Abstimmung mit anderen, wohingegen im MAS Kooperation oder Konkurrenz zwischen den Agenten erforderlich wird. Dies spiegelt sich in der Komplexität wider. Single-Agenten-Systeme sind einfacher zu entwerfen und zu steuern, MAS weisen aufgrund der Interaktionen und gegenseitigen Abhängigkeiten eine höhere Komplexität auf. Diesem höheren Gestaltungsaufwand stehen jedoch deutliche Vorteile des MAS gegenüber. Bei der Skalierbarkeit ist das Single-Agenten-System begrenzt, sodass die Lösung großer Probleme herausfordernd sein kann, während ein MAS Aufgaben auf mehrere Agenten verteilt und dadurch gut skaliert. Ähnlich verhält es sich mit der Fehlertoleranz. Fällt im Single-Agenten-System der eine Agent aus, kann das gesamte System versagen. Ein MAS bleibt robuster, da der Ausfall eines einzelnen Agenten die übrigen nicht zwangsläufig beeinträchtigt. Eng damit verknüpft sind Kommunikation und Koordination. Ein Single-Agenten-System benötigt weder eine Kommunikation zwischen Agenten noch Koordinationsmechanismen. Im MAS hingegen tauschen die Agenten Informationen aus oder verhandeln miteinander, weshalb explizite Strategien wie Kooperation, Wettbewerb oder Verhandlung notwendig werden. Auch die Interaktion mit der Umgebung unterscheidet sich. Der einzelne Agent agiert isoliert oder tritt nur minimal mit seiner Umgebung in Kontakt, während MAS aktiv und häufig in geteilten Umgebungen agieren (Huang, 2025, S. 53). Die Zusammenarbeit in einem MAS lässt sich in drei verschiedene Arten zusammenfassen. In einem kollaborativen System verfolgen alle Agenten ein gemeinsames Ziel. Die Agenten müssen hierbei nicht nur ihre Aufgaben erledigen, sondern auch ihre Informationen mit den anderen Agenten teilen. Gegensätzlich dazu steht ein kompetitives System. Darin besitzen die Agenten gegensätzliche Ziele und sie müssen nicht nur ihr Ziel durchsetzen, sondern auch Handlungen anderer antizipieren, wie in Sicherheitsanwendungen, wobei Systeme sich vor Angreifern schützen müssen (Huang, 2025, S. 55–57).

Darüber hinaus gibt eine vielzitierte Bestimmung von Harrison Chase die Definition eines generativen KI-Agenten an: Demnach liegt ein generativer KI-Agent dann vor, wenn ein LLM den Steuerungsfluss einer Anwendung bestimmt, also selbst entscheidet, welche Aktionen in welcher Reihenfolge ausgeführt werden, statt einer fest vorgegebenen Abfolge zu folgen (Chase, zitiert nach Taulli & Deshmukh, 2025, S. 4–5). Beschreiben lassen sich generative KI-Agenten über ihre Komponenten, die je nach Anwendungsfall ausgewählt werden. Die Reflexion bezeichnet die Fähigkeit, die eigenen kognitiven Prozesse zu prüfen und anzupassen, etwa um Entscheidungen zu hinterfragen und Schlussfolgerungen besser zu begründen. Die Nutzung von „Tools“ erlaubt die Interaktion mit externen Werkzeugen, Programmierschnittstellen (APIs) oder Software, um etwa aktuelle Informationen abzurufen oder Abläufe zu automatisieren. Das „Memory“ speichert Informationen aus früheren Interaktionen und gliedert sich neben einem Kurzzeitgedächtnis in ein meist über Vektordatenbanken realisiertes Langzeitgedächtnis mit episodischem, semantischem und prozeduralem Anteil. Das „Planning“ nutzt LLMs, um eigenständig Schritte zur Zielerreichung zu bestimmen und komplexe Vorhaben in Teilaufgaben zu zerlegen. Die Multi-Agenten-Kollaboration lässt mehrere LLMs, ähnlich wie in einem Team, arbeitsteilig zusammenwirken, was einzelnen Agenten häufig überlegen ist. Die Autonomie schließlich umfasst Entscheidungen ohne fortlaufende menschliche Eingriffe, wobei zwischen Autonomie und Kontrolle abzuwägen ist und menschliche Aufsicht unverzichtbar bleibt (Taulli & Deshmukh, 2025, S. 5–12).

Die Funktionsweise generativer KI-Agenten lässt sich anhand der Sieben-Schichten-KI-Agenten Architektur nach Agentic AI strukturiert beschreiben. Dieses Modell teilt das Ökosystem in Ebenen auf, die jeweils eine eigene Aufgabe übernehmen. Höhere Schichten greifen dabei auf die Funktionen

der unteren zu, ohne deren Details zu kennen. Dies stellt zudem eine einheitliche Strukturierungsgrundlage bereit, um KI-Agenten systematisch zu entwickeln.

1. Foundation Models

Die unterste Ebene bildet die kognitive Grundlage des Agenten. Sie umfasst LLMs wie GPT-4, Claude oder Gemini, die das Reasoning, natürliches Sprachverständnis und multimodale Verarbeitung von Text, Bildern und strukturierten Daten bereitstellen.

2. Data Operations

Diese Schicht verwaltet die Dateninfrastruktur des Agenten. Vektordatenbanken ermöglichen Ähnlichkeitssuchen und semantisches Matching, sogenannte „ETL-Pipelines" übernehmen die Reinigung, Transformation und Anreicherung heterogener Datenquellen. Mit „Agentic RAG" orchestrieren Agenten zudem die Informationsbeschaffung und -verarbeitung selbst, um die Genauigkeit der Ausgaben zu erhöhen.

3. Agent Frameworks

Software-Frameworks wie LangChain oder AutoGen orchestrieren den Denkprozess des Agenten. Sie steuern das Prompt-Management, das Gedächtnis und die Zustandsverwaltung und ermöglichen die Erstellung komplexer Arbeitsabläufen sowie die Integration externer Tools.

4. Deployment and Infrastructure

Diese Ebene stellt die technische Grundlage für den skalierbaren Betrieb bereit. Cloud-Plattformen liefern Rechenleistung und Speicher, Container-Orchestrierung über Kubernetes verwaltet Deployment und Fehlertoleranz, und sogenannte „CI/CD-Pipelines" automatisieren Tests sowie Updates.

5. Evaluation and Observability

Zur Qualitätssicherung autonomer Systeme dienen Metriken zu Korrektheit, Relevanz, Robustheit und Alignment mit ethischen Leitlinien. Observability-Plattformen wie z. B. LangSmith oder AgentOps.ai protokollieren Interaktionen in Echtzeit zur Fehleranalyse und Leistungsoptimierung.

6. Security and Compliance

Sicherheit und Compliance stellen formal eine eigene Schicht dar, wirken jedoch als vertikale Disziplin in alle übrigen Ebenen hinein. Inhaltlich umfasst die Schicht den Schutz vor Datenlecks, Angriffen und unbefugtem Zugriff sowie die Einhaltung regulatorischer Vorgaben.

7. Agent Ecosystem (Anwendungsschicht)

Die höchste Ebene überführt die technischen Fähigkeiten in konkreten Geschäftsnutzen. Die Anwendungsfelder reichen von Kundenservice-Plattformen über automatisierte Inhaltsgenerierung bis hin zu branchenspezifischen Lösungen, etwa im Versicherungs- oder Bankensektor. Die Integration erfolgt typischerweise über bestehende Systeme (Huang, 2025, S. 24–34).

3.5 Grenzen und Sicherheit Künstlicher Intelligenz

Der Einsatz von KI ist jedoch mit Vorsicht zu genießen, denn KI-Systeme stoßen regelmäßig an Grenzen. So haben sie Probleme, wenn sie auf neue, unerwartete Situationen reagieren müssen. Menschen können dabei kreativ reagieren, während es im Bereich Künstlicher Intelligenz zwar kreative generative Deep-Learning-Systeme gibt, die sich dort aber noch weiterentwickeln müssen. Zudem gibt es noch kein erfolgversprechendes Modell zur Implementierung von Bewusstsein oder intrinsischer Motivation. Dadurch können KI-Systeme noch nicht auf verschiedensten Ebenen über sich selbst reflektieren (Ertel, 2016, S. 15). Des Weiteren arbeiten Large Language Models sehr unzuverlässig bei streng formalen Aufgaben, wie Mathematik, da neuronale Netze grundsätzlich Schwierig-

keiten mit exakter Logik haben (Ertel, 2016, S. 74). Hinzu kommt, dass manche KI-Systeme hochkomplex sind, sodass auch Menschen diese nicht mehr wirklich durchschauen. Die Ursache eines Fehlers lässt sich dabei oft nicht identifizieren, was Auswirkungen auf die Sicherheitsbewertung hat, denn in anderen Bereichen versuchen Spezialisten Systeme zu verstehen und prüfen diese anhand von z. B. Checklisten. Dies wird bei Künstlicher Intelligenz nicht mehr in vollem Ausmaß der Fall sein (Ertel, 2016, S. 358). Eine weitere Limitation bzw. sogar Problem ist die Halluzination. Die Halluzination beschreibt das Erzeugen von falschen oder irreführenden Informationen, was bei Modellen mit Transformer-Architektur der Fall ist. Zwar haben sich LLMs in ihrer Zuverlässigkeit verbessert, jedoch muss noch viel Arbeit eingesetzt werden, um sie in anspruchsvollen Umgebungen zuverlässiger zu machen. Zusätzlich dazu sind sie ressourcenintensiv, da sie viel Rechenleistung und Energie beanspruchen, dadurch sind sie teuer und auch im großen Maßstab wenig nachhaltig (Taulli & Deshmukh, 2025, S. 256). Über die Grenzen der KI-Systeme hinaus gibt es auch erhebliche Schwachstellen im Hinblick auf Sicherheit. Der OWASP-Leitfaden "OWASP Top 10 for LLM Applications 2025" ist eine von Fachleuten erstellte Prioritätenliste, welche die zehn aktuellen zentralen und kritischen Schwachstellen von LLM-Anwendungen benennt und erklärt. Die dabei am höchsten priorisierte Schwachstelle ist, wenn Benutzereingaben das Verhalten des LLM auf unbeabsichtigte Weise verändern. Die Eingabe kann dabei entweder direkt das Verhalten des Modells verändern, was willentlich oder unwissend des Nutzers geschehen kann, oder das Modell nimmt externe Dateien während der Verarbeitung auf und verändert dadurch sein Verhalten. Folgen können sein, dass Modelle z. B. Richtlinien verletzen, schädliche Inhalte erzeugen oder unbefugten Zutritt ermöglichen (OWASP, 2024, S. 3–4). Die unbeabsichtigte Offenlegung vertraulicher Daten wird danach priorisiert. Dabei können LLMs vertrauliche Daten, wie Finanz- und Gesundheitsdaten oder Zugangsdaten ungewollt in ihren Antworten preisgeben. Es wird davor gewarnt, sensible Daten den LLMs zur Verfügung zu stellen, denn sie könnten diese benutzen und in ihren Antworten preisgeben (OWASP, 2024, S. 7).

3.6 Stand der Forschung zu generativen KI-Agenten im Forecasting von SCM

Die Anwendung von KI und maschinellem Lernen im SCM ist als Forschungsfeld breit etabliert. Ferreira, Francisco und Pinho (2025) belegen in einem systematischen Literaturüberblick über 109 Studien, dass die einschlägige Literatur in den vergangenen Jahren stark gewachsen ist und ein breites Anwendungsspektrum abdeckt, darunter Nachfrageprognose, Bestandsmanagement und Risikoanalyse (Ferreira, Francisco & De Pinho, 2025, S. 157828). Generative KI sowie große Sprach- und multimodale Modelle nennen die Autoren dabei ausdrücklich als mögliche künftige Forschungsrichtung (Ferreira et al., 2025, S. 157836–157837). Auf ein ähnliches Ergebnis schließen Richey Jr, Chowdhury, Davis-Sramek, Giannakis, & Dwivedi (2023). Sie untersuchen das Potenzial von generativer KI für die Logistik und das SCM und schlagen einen Forschungsrahmen für die Zukunft vor (Richey, Chowdhury, Davis-Sramek, Giannakis & Dwivedi, 2023, S. 532). Dabei würde generative KI enormes Potenzial für die gesamte Supply Chain bergen (Richey et al., 2023, S. 540), jedoch können traditionelle Theorien die Komplexität der Veränderungen durch KI nicht mehr abbilden (Richey et al., 2023, S. 544). Daher schlagen sie ein Forschungsframework vor, anhand dessen die KI die Logistik und SCM transformieren kann (Richey et al., 2023, S. 539). Hendriksen (2023) argumentiert, dass zukünftige Forschung neue theoretische Begriffe entwickeln muss, weil KI nicht mehr nur als passives Tool verstanden werden kann, sondern teilweise als autonomer oder mitentscheidender Akteur (Hendriksen, 2023, S. 72). Er hat daher selbst ein Framework zur Integration von KI in Supply Chains entwickelt (Hendriksen, 2023, S. 70).

Ein eigenes Forschungsfeld bilden agentenbasierte Systeme. In ihrem Literaturüberblick stellen Xu, Mak und Brintrup (2021) fest, dass agentenbasierte Technologie inzwischen ausgereift ist (Xu et al., 2021, S. 1), argumentieren aber zugleich, dass es an allgemein akzeptierten Leistungskennzahlen und

Benchmark-Problemen fehlt (Xu et al., 2021, S. 10–11) und dass Entwicklungen des maschinellen Lernens und der prädiktiven Analytik bislang keinen Eingang in die agentenbasierte SCM-Forschung gefunden haben (Xu et al., 2021, S. 14). Generative KI oder LLMs werden dabei nicht betrachtet. Diese Beobachtung bestätigt die spätere Implementierungsstudie von Xu, Mak, Minaricova und Brintrup (2024): Ihr Multi-Agenten-Prototyp einer autonomen Fleischlieferkette stützt sich auf eine regelbasierte und vergleichsweise einfache Entscheidungslogik. Dieser Prototyp würde zwar erfolgreich automatisierte Funktionen integrieren und Entscheidungspunkte verknüpfen, jedoch fehlt es ihm an der Fähigkeit zur Neukonfiguration und Anpassung an Störungen, was für autonome Supply Chains wichtig zur Erreichung von Resilienz ist. Auch Planungsfunktionen, welche entscheidend für SCM sind, fehlen. LLMs wie ChatGPT und Llama könnten es hier erlauben, KI-Agenten zu erstellen, die tatsächlich in der Lage sind, Entscheidungen zu treffen und zu planen (Xu, Mak, Minaricova & Brintrup, 2024, S. 13–14).

Trotz des jungen Forschungsfeldes von generativer KI im Forecasting oder SCM-Kontext gibt es bereits Anwendungen. Eine in die andere Richtung gedachte Integrationsmöglichkeit von KI in SCM bieten Jackson, Ivanov, Dolgui und Namdar (2024) mit ihrem Framework. Sie ordnen dabei Kernkompetenzen von KI und generativer KI über ein fähigkeitsbasiertes Framework 13 Operationen im SCM zu, darunter auch dem Forecasting. Das Forecasting fällt dabei in die Operation mit den meisten Überschneidungen (Jackson, Ivanov, Dolgui & Namdar, 2024, S. 6127). Die Bestätigung dafür, dass Forecasting zu den Kompetenzen der KI passt, liefern Panja, Zheng und Glock (2026). Sie werten 43 Arbeiten aus und identifizieren Forecasting mit neun Studien als das am stärksten bearbeitete Einsatzfeld generativer KI im SCM. Dabei wird generative KI vor allem dort als nützlich beschrieben, wo Echtzeitdaten fehlen oder die Datenbasis zu klein ist. Dann kann sie aus historischen Daten Verteilungen lernen und synthetische Daten erzeugen, die anschließend für Prognosen genutzt werden (Panja et al., 2026, S. 10). Außerdem kann generative KI im Forecasting nicht nur Daten ergänzen, sondern auch Datenqualität und Planung verbessern. Für Planning und Forecasting wird beschrieben, dass generative KI große heterogene Datensätze filtern, normalisieren und Fehler korrigieren kann, und dass seltene, aber geschäftskritische Ereignisse wie Promotions, Stockouts, Wetterschocks oder Lieferantenvorfälle synthetisch erzeugt werden können, um Startprobleme oder Ungleichgewichtsprobleme in der Nachfrage- und Absatzplanung zu mindern (Panja et al., 2026, S. 20). Empirisch belegen zwei Befragungsstudien aus dem chinesischen Fertigungskontext eine positive Wirkung generativer KI auf die Leistung von Supply Chains. Li, Zhu, Chen und Liu (2024) zeigen auf Basis von 213 chinesischen Fertigungsunternehmen, dass generative KI-Nutzung positiv mit nachhaltiger Supply-Chain-Performance zusammenhängt (Li, Zhu, Chen & Liu, 2024, S. 1). Dieser Zusammenhang wird über grüne Supply-Chain-Kollaboration und Circular-Economy-Implementierung vermittelt (Li, Zhu, et al., 2024, S. 7–8), sodass sich die Vorteile generativer KI erst durch ihre Einbettung in nachhaltige Supply-Chain-Praktiken voll entfalten (Li, Zhu, et al., 2024, S. 9). Li, Liu, Jin, Cheng und Zhang (2024) bestätigen anhand von 236 Unternehmen einen positiven Zusammenhang zwischen der Nutzungstiefe generativer KI und der Supply-Chain-Performance, der über Lieferanten- und Käuferkoordination vermittelt wird (Li, Liu, Jin, Cheng & Zhang, 2024, S. 8). Auffällig ist dabei, dass Supply-Chain-Dynamik diese Mediationseffekte verstärkt (Li, Liu, et al., 2024, S. 6), der koordinationsbezogene Nutzen generativer KI also gerade in volatilen Umfeldern größer ist (Li, Liu, et al., 2024, S. 7). Zouaghi, Beldjoudi, Gunasekaran, Fosso Wamba und Sahrane (2025) belegen in einer Studie die positive Wirkung von generativer KI auf das Forecasting. Sie untersuchen dabei, wie die Einstellung der Temperatur von generativer KI Entscheidungen im SCM optimieren kann (Zouaghi, Beldjoudi, Gunasekaran, Fosso Wamba & Sahrane, 2025, S. 1). Die Temperatureinstellung ist ein Parameter zur Anpassung der Wahrscheinlichkeitsverteilung von Modelloutputs (Zouaghi et al., 2025, S. 6). Innerhalb des Forecastings wird dabei gezeigt, dass das KI-Modell bei niedrigen Temperatureinstellungen die besten Fehlerwerte erzielt und das Prognosemodell ARIMA übertrifft (Zouaghi et al., 2025, S. 21).

Erst jüngste Arbeiten verbinden generative KI mit Multi-Agenten-Systeme. Quan, Liu, Benaben, & Montreuil (2026) schlagen mit MARS ein LLM-basiertes Multi-Agenten-Framework für die Risikobewertung in Supply-Chain-Netzwerken vor, welches aus drei Agenten sowie einer iterativen Feedback-Revisions-Schleife besteht (Quan et al., 2026, S. 6). Im Vergleich zu einer regelbasierten Baseline steigt die Übereinstimmung mit menschlichen Bewertungen deutlich (Quan et al., 2026, S. 21–22). Gegenstand ist jedoch die Supply-Chain-Risikobewertung zur Standortbestimmung, nicht jedoch das Forecasting (Quan et al., 2026, S. 5). Am nächsten an dieser Arbeit liegt die Studie von Jannelli, Schoepf, Bickel, Netland, & Brintrup (2026). Sie untersuchen ein kognitiv inspiriertes, modulares Multi-Agenten-Framework für sequenzielle Supply Chains, in dem LLM-Agenten Umweltinformationen wahrnehmen, in Speicherstrukturen ablegen, Entscheidungen treffen, mit Nachbaragenten kommunizieren und diese schließlich ausführen (Jannelli et al., 2026, S. 9–10). Die empirische Evaluation erfolgt durch zwei LLMs anhand zweier Zielgrößen im Bestandsmanagement (Jannelli et al., 2026, S. 14). Die Agenten werden hierbei verschieden konfiguriert, ob alleine, mit oder ohne Tool, mit oder ohne Informationsaustausch/Verhandlung zu anderen Agenten (Jannelli et al., 2026, S. 9). Das Forecasting ist hier als lineares Regressions-Tool in die Agenten eingebettet (Jannelli et al., 2026, S. 10). Ergebnisse zeigen, dass die Konfigurationen mit Informationsaustausch und Verhandlungen und die Nutzung von Tools hinsichtlich der Erreichung der Zielgrößen am effektivsten sind (Jannelli et al., 2026, S. 29).

3.7 Forschungslücke

Aus dem dargestellten Forschungsstand ergibt sich ein Bild entlang drei Richtungen. Erstens ist generative KI im Forecasting als Anwendungsfeld gut belegt: Sie ist das am stärksten bearbeitete Einsatzfeld generativer KI im SCM (Panja et al., 2026, S. 8), verbessert Datengrundlage und Planung (Panja et al., 2026, S. 10, S. 20), übertrifft bei geeigneter Konfiguration klassische Prognosemodelle wie ARIMA (Zouaghi et al., 2025, S. 21) und wirkt sich positiv auf die Supply-Chain-Performance aus (Li, Liu, et al., 2024, S. 8; Li, Zhu, et al., 2024, S. 7–8). Sämtliche dieser Arbeiten untersuchen generative KI jedoch nicht als autonomes Agentensystem. Zweitens existiert mit der agentenbasierten SCM-Forschung zwar eine ausgereifte Tradition zur Architektur autonomer Systeme (Xu et al., 2021, S. 1), doch bleibt diese durchgängig nicht-generativ: Entwicklungen des maschinellen Lernens und der prädiktiven Analytik haben dort bislang keinen Eingang gefunden (Xu et al., 2021, S. 14), und auch der spätere Implementierungsprototyp stützt sich auf regelbasierte Entscheidungslogik ohne Planungsfunktionen (Xu et al., 2024, S. 13). Das Fundament für Agentensysteme ist somit vorhanden, nicht aber dessen Verbindung mit generativer KI. Drittens existieren erst jüngst LLM-basierte Multi-Agenten-Systeme, die beides verbinden, deren Untersuchungsgegenstand jedoch die Risikobewertung (Quan et al., 2026, S. 5) oder das Bestandsmanagement (Jannelli et al., 2026, S. 14) ist. Selbst in der thematisch nächsten Arbeit von Jannelli et al. (2026) ist die Bedarfsprognose lediglich als lineares Regressions-Tool in die Agenten eingebettet (Jannelli et al., 2026, S. 10) und nicht eigenständiger Gegenstand der Evaluation.

Damit verbindet keine der vorliegenden Arbeiten LLM-basierte Multi-Agenten-Systeme mit der Erstellung von Prognosen über die Nachfrage als zentralem Untersuchungsgegenstand. Diese Lücke wird von den Autoren des Feldes selbst gestützt, und zwar aus zwei entgegengesetzten Richtungen: Aus der generativen KI-Forschung heraus benennen Zouaghi et al. (2025) adaptive Multi-Agenten-Systeme ausdrücklich als künftigen Integrationspfad (Zouaghi et al., 2025, S. 34), während aus der Agentenforschung heraus Xu et al. (2024) LLMs wie ChatGPT und Llama als Weg benennen, Agenten mit echter Entscheidungs- und Planungsfähigkeit auszustatten (Xu et al., 2024, S. 14). Beide Forschungsstränge weisen damit auf dieselbe Verbindung hin: ein LLM-basiertes Multi-Agenten-System, welches für das Forecasting bislang jedoch nicht umgesetzt wurde. Verstärkt wird diese Aussage dadurch, dass ein erheblicher Teil der Literatur zu generativer KI im SCM konzeptioneller Natur ist und Frameworks oder Forschungsagenden vorschlägt (Hendriksen, 2023, S. 70; Jackson et al.,

2024, S. 6126; Richey et al., 2023, S. 539), während empirische Evidenz im Schnittfeld von generativer KI, Agentensystemen und Forecasting faktisch auf eine einzige Arbeit beschränkt bleibt (Jannelli et al., 2026, S. 1), und diese adressiert das Forecasting nur als externe Integration. An dieser Stelle setzt die vorliegende Arbeit an: Sie konzipiert und evaluiert ein LLM-basiertes Multi-Agenten-System, dessen zentraler Gegenstand das Forecasting im SCM ist.

4. Resultate

4.1 Ergebnisse FF1: Anforderungsliste und Kompetenzprofil

Auf Basis der Sieben-Schichten-KI-Agenten-Architektur (beschrieben im Kapitel 3.4), ergibt sich ein Gerüst für eine Liste technologischer Anforderungen an KI-Agenten. Die Funktionsweise generativer KI-Agenten enthält dabei folgende Ebenen: Foundation Models, Data Operations, Agent Frameworks, Deployment & Infrastructure, Evaluation & Observability, Security & Compliance, Agent Ecosystem (Huang, 2025, S. 24–34). Mit der Abfolge des Forecasting-Prozesses ergibt sich ein Gerüst an Anforderungen mit Bezug auf das Forecasting im SCM. Der Forecasting-Prozess ergibt sich folgendermaßen anhand von Kapitel 3.2:

1. Datenerfassung und -aufbereitung

Forecasting beruht auf der Annahme, dass erkennbare Muster in historischen Daten eine Vorhersage zukünftiger Werte ermöglichen (Petropoulos et al., 2022, S. 710–711)

2. Analyse der Zeitreihenkomponenten

Zerlegung: Beobachtete Nachfrage = systematische Komponente + zufällige Komponente, bestehend aus Niveau, Trend und Saisonalität. Und mit dem Ziel, die zufällige Komponente herauszufiltern und die systematische zu schätzen (Chopra & Meindl, 2019, S. 189).

3. Methodenwahl

Einteilung der Methoden in vier Hauptgruppen: Qualitative, Zeitreihen-, Kausale und Simulationsverfahren samt jeweiliger Eignungsbedingungen (Chopra & Meindl, 2019, S. 189). Und innerhalb der Zeitreihenverfahren gibt es verschiedene Anwendungsvoraussetzungen der einzelnen Modelle, z. B. Einfache Exponentielle Glättung nur ohne Trend/Saisonalität (Chopra & Meindl, 2019, S. 198–202).

4. Prognoseerstellung

Prognoseformen: Punktprognose, Prognoseintervall, Perzentil, Prognoseverteilung(Petropoulos et al., 2022, S. 710–711). Verschiedene Methoden, wie Holts Modell oder Winters Modell (Chopra & Meindl, 2019, S. 198–202).

5. Bestimmung des Prognosefehlers

Prognosen sind immer falsch und sollten eine Fehlermessung beinhalten (Chopra & Meindl, 2019, S. 187). Fehlermaße-Beispiele: MSE und MAD (Chopra & Meindl, 2019, S. 202).

6. Integration in die Bedarfsplanung

Nutzung der Prognose für nachgelagerte SCM-Entscheidungen, da sowohl für Push-, als auch für Pull-Prozesse eine Prognose essentiell ist (Chopra & Meindl, 2019, S. 186).

Die einzelnen Schritte des Forecasting-Prozesses lassen sich folgendermaßen in das Sieben-Schichten-Modell einfügen. Die Datenerfassung und -aufbereitung ist Data Operations zuzuordnen, da diese die Dateninfrastruktur des Agenten darstellt. Die Analyse der Zeitreihenkomponenten, Methodenwahl, Prognoseerstellung und Bestimmung des Prognosefehlers ist der Schicht Agent Frameworks zuzuordnen, da hier der Denkprozess des Agenten orchestriert wird. Die Integration in die

Bedarfsplanung lässt sich dem Agent Ecosystem zuordnen, da hier technische Fähigkeiten in konkreten Geschäftsnutzen umgewandelt werden durch Systeme. Die restlichen Schichten lassen sich übergreifend auf den gesamten Forecasting-Prozess anwenden, da diese grundlegenden Anforderungen, wie das LLM als kognitive Basis oder Sicherheit während der Ausführung behandeln.

Nachfolgend werden die Ergebnisse der Literaturrecherche für FF1 konzeptorientiert auf dieses Gerüst angewendet. Die Anforderungen werden dabei entlang der Architekturschichten des Sieben-Schichten-Modells angewendet. Als theoriegeleitet wird dabei die Ableitung einer Anforderung aus der Schicht des Sieben-Schichten-Modells oder dem Schritt im Forecasting dargelegt. Literaturbeständig wird es, sobald die Literatur dies bestätigt und literaturneu ergibt sich, wenn die Literatur eine neue Anforderung darlegt. Die abschließende Tabelle fasst den Katalog zusammen.

Foundation Models: Theoriegeleitet bildet ein leistungsfähiges LLM mit Sprachverständnis und Schlussfolgern die kognitive Basis. Die Literatur bestätigt dies, wobei LLM-basierte Agenten im Hinblick auf die Erreichung zweier Zielgrößen im SCM getestet wurden und LLM-Agenteninteraktionen mit einem hohen Grad an Koordination sogar die Leistungen mathematisch gesteuerter Werkzeuge übertreffen (Jannelli et al., 2026, S. 15). Darüber wird gezeigt: Strukturierte Schritt-für-Schritt-Prompts bei LLMs senken bei Vergleichen in der Prognoseerstellung im Forecasting den Prognosefehler deutlich gegenüber einfachen Prompts und verbessern die Schlussfolgerstruktur des Modells (Abbaszadeh Nakhost, Bloemer & Minner, 2025, S. 12).

Data Operations: Theoriegeleitet erfordert das Forecasting eine Dateninfrastruktur, die vollständige, konsistente Nachfragezeitreihen bereitstellt. Die Literatur bestätigt die Datenqualität als zentrale Voraussetzung, da inkonsistente oder fehlende Daten die Verlässlichkeit der Agenten-Outputs unmittelbar kompromittieren können (Trifone, Dallari & Brintrup, 2026, S. 21). In der Multi-Agenten-Architektur von Shukla und Kiridena (2016) sind Datenabruf und Vorverarbeitung zudem als eigenständige Agentenfunktionen realisiert, was die theoriegeleitete Verankerung der Datenaufbereitung in der Agentenarchitektur stützt (Shukla & Kiridena, 2016, S. 6990). Bestätigt wird zudem die Anbindung externer Daten- und Wissensquellen in Echtzeit, um Wissensgrenzen des Modells abzumildern und Nützlichkeit und Genauigkeit des Modells zu verbessern (Feuerriegel, Hartmann, Janiesch & Zschech, 2024, S. 5–6).

Agent Frameworks: Theoriegeleitet führt diese Schicht die Agentenkomponenten Planning, Memory, Reflexion und Tool-Nutzung aus. Aus der Unzuverlässigkeit von LLMs bei streng formalen Aufgaben folgt zudem die Anforderung der Nutzung von Tools bei Deterministik, also der Trennung von sprachlichem Schlussfolgern und deterministischer Berechnung. Es wird gezeigt, dass beim Prognostizieren der zukünftigen Nachfrage und Minimierung der Kosten ein geeignetes Tool und eine hohe Gewichtung des Tools, ohne zusätzliches Konsens-Findungs-Framework die beste Leistung erbringen. Zudem wird gesagt, dass das Grundmodell (LLM) in der Lage ist, einfache Berechnungen durchzuführen, da es sich um eine probabilistische Maschine handelt und nicht um ein Tool, das für komplexe Berechnungen geeignet ist (Jannelli et al., 2026). Dies erfährt auch Bestätigung im Artefakt von Abbaszadeh Nakhost, Bloemer und Minner (2025). Dort erzeugt das LLM ausführbaren Code, der lokal deterministisch ausgeführt wird, statt Prognosewerte frei zu generieren (Abbaszadeh Nakhost et al., 2025, S. 6–7). Außerdem würden Experimente zur globalen Kostenminimierung zeigen, dass der Einsatz eines Tools nur dann effektiv wäre, wenn die dem Tool zugrunde liegende Funktion mathematisch in der Lage ist, eine bestimmte Herausforderung zu bewältigen (Jannelli et al., 2026, S. 19). Theoriegeleitet bleibt zudem die Anforderung, Zeitreihenmuster anhand erklärbarer Kennzahlen zu diagnostizieren und daraufhin eine Forecasting-Methode zu wählen. Eine Bestätigung erfährt dies durch einen Vergleich verschiedener Forecasting-Methoden. Dabei verbessert eine datengetriebene Selektion anhand von Zeitreihencharakteristika die Prognosegüte gegenüber einer festen Klassifikation oder Einheitsmethode (Abbaszadeh Nakhost et al., 2025, S. 12–13).

Deployment und Infrastructure: Theoriegeleitet stellt diese Schicht den skalierbaren Betrieb über Cloud- und Container-Infrastrukturen sicher. Bestätigung erhält dies in einer Studie von Dincelli & Jafarian (2025), wobei die Autoren einen KI-gestützten Trainingsprototypen in einer metaverse-ähnlichen Umgebung entwickelten und ihn von 53 Cybersicherheitsfachleuten beurteilen ließen. Auf dieser Basis analysieren sie Potenziale, Herausforderungen und praktische Implikationen (Dincelli & Jafarian, 2025, S. 318–320). Sie fordern dabei eine skalierbare und interoperable Infrastruktur, die Verbindung von Edge- und Cloud-Computing, die Containerisierung von Services sowie die Nutzung standardisierter APIs und Middleware (Dincelli & Jafarian, 2025, S. 325).

Evaluation and Observability: Theoriegeleitet dienen Metriken zu Korrektheit und Robustheit sowie Überwachungs-Werkzeuge der Qualitätssicherung. Die Literatur bestätigt dies nur mit der Feststellung, dass viele Studien auf idealisierten oder simulierten Datensätzen beruhen und Vergleiche mit traditionellen statistischen oder hybriden KI-Verfahren selten bleiben, sodass der marginale Mehrwert von LLMs ungeklärt ist. Es wird daher gefordert, über Proof-of-Concept-Implementierungen hinauszugehen und robuste, vergleichende Evaluationsrahmen zu entwickeln (Song, Xie, Yang & Zhao, 2026, S. 13). Hendriksen (2023) fordert ergänzend robuste Systeme zum laufenden Monitoring der KI-Leistung samt Vorkehrungen zur Störungsbewältigung und Notfallplänen (Hendriksen, 2023, S. 73). Für agentenbasierte Systeme im Supply Chain Management wird neben hoher Zuverlässigkeit explizit die Implementierung von Fallback-Strategien gefordert, die gewährleisten, dass der Ausfall eines einzelnen Agenten nicht auf andere Agenten übergreift (Xu et al., 2021, S. 14). Daraus lässt sich ableiten, dass Robustheits- und Rückfallmechanismen eine zentrale, bislang unzureichend adressierte Designanforderung für LLM-basierte Agentensysteme in der Prognose darstellen. Literaturneu ist die Kontrolle der inhärenten Output-Stochastizität: Abbaszadeh Nakhost et al. (2025) zeigen, dass erst die Mittelung über wiederholte LLM-Läufe im Ansatz GPT-Robust die inhärente Output-Stochastizität wirksam reduziert, Prognosefehler auf Benchmark-Niveau ermöglicht und zugleich die Volatilität einzelner Läufe vermeidet (Abbaszadeh Nakhost et al., 2025, S. 11–12).

Security and Compliance: Theoriegeleitet umfasst diese Schicht den Schutz vor Datenlecks und Angriffen sowie regulatorische Konformität. Der Schutz vor Prompt Injection, Datenexfiltration und unautorisiertem Logging beim Einsatz externer oder cloudbasierter Agenten, wird in den Limitationen von Jannelli et al. (2026) beschrieben, allerdings mit Verweis auf eine weitere Quelle: Carlini et al. (2021) zeigen, dass große Sprachmodelle selbst seltene Trainingsdaten wörtlich memorisieren und per Black-Box-Abfragen extrahierbar machen, einschließlich personenbezogener Informationen, wobei größere Modelle dafür besonders anfällig sind (Carlini, Tramer, Wallace, Jagielski, Herbert-Voss, Lee et al., 2021, S. 2640–2642).

Agent Ecosystem: Theoriegeleitet überführt die Anwendungsschicht die technischen Fähigkeiten in Geschäftsnutzen, verlangt Abwägung zwischen Autonomie und Kontrolle und erfolgt über bestehende Systeme. Auch für den SCM-Kontext ist die Interoperabilität mit bestehenden Planungssystemen und Supply-Chain-Informationssystemen zentral, da Xu et al. (2021) deren mangelnde Ausprägung als Adoptionshemmnis agentenbasierter Systeme beschreiben (Xu et al., 2021, S. 1, S. 14). Jedoch muss dabei auch betrachtet werden, wie Trifone, Dallari und Brintrup (2026) in ihren Interviews zeigen, dass ein klarer Konsens zur Wichtigkeit des Human-in-the-loop besteht und dass die Befragten es ablehnen, KI-Agenten bei Entscheidungen mit erheblichem Einfluss auf das SCM mit voller Entscheidungsautonomie auszustatten (Trifone et al., 2026, S. 23–24). Jannelli et al. (2026) betonen, dass reale operative Entscheidungen weiterhin einen Human-in-the-Loop erfordern, denn es müsste jemanden geben, der die Verantwortung trägt, was bedeutet, dass ein Mensch die Maßnahmen genehmigen und bei Bedarf eingreifen muss (Jannelli et al., 2026, S. 18). Sie begründen dies damit, dass die Outputs noch nicht vollständig verlässlich und erklärbar sind (Jannelli et al., 2026, S. 21).

Das theoretische Grundgerüst mit der Erweiterung der Literatur ergibt nun folgende Liste an technologischen Anforderungen an KI-Agenten im SCM-Kontext. Die Nummer dient lediglich zum Überblick der Anforderungen und orientiert sich an der Abfolge der zuvor im Text genannten Anforderungen und stellt keine Wertung dar. Die Anforderung stellt die im Text dargestellte Anforderung dar. Die Schicht stellt den Bezug zur Ebene im Sieben-Schichten-Modell dar. Der Prozessbezug stellt den Schritt im benannten Forecasting-Prozess dar. Die Herkunft gibt an, ob die Anforderung theoriegeleitet (T), literaturbeständig (LB) oder literaturneu (LN) abgeleitet wurde. Als Quelle(n) wird die Literatur benannt, die die jeweilige Anforderung zusätzlich zu den theoretischen Grundlagen belegt.

Nr.	Anforderung	Schicht	Prozessbezug	Herkunft	Quelle(n)
1	Leistungsfähiges LLM als kognitive Basis	Foundation Models	übergreifend	T / LB	(Jannelli et al., 2026, S. 15)
2	Strukturierte Schritt-für-Schritt-Prompts	Foundation Models	übergreifend	LB	(Abbaszadeh Nakhost et al., 2025, S. 12)
3	Datenqualität: vollständige, konsistente Nachfragezeitreihen	Data Operations	Datenerfassung/-aufbereitung	T / LB	(Trifone et al., 2026, S. 21)
4	Datenabruf und Vorverarbeitung als eigenständige Agentenfunktionen	Data Operations	Datenerfassung/-aufbereitung	T / LB	(Shukla & Kiridena, 2016, S. 6990)
5	Echtzeit-Anbindung externer Daten- und Wissensquellen	Data Operations	Datenerfassung/-aufbereitung	T / LB	(Feuerriegel et al., 2024, S. 5–6)
6	Tool-Grounding	Agent Frameworks	übergreifend	T / LB	(Abbaszadeh Nakhost et al., 2025, S. 6–7; Jannelli et al., 2026, S. 15)
7	Mathematische Passung des Tools	Agent Frameworks	übergreifend	LN	(Jannelli et al., 2026, S. 19)
8	Erklärbare Diagnose der Zeitreihenmuster als Grundlage der Modellwahl	Agent Frameworks	Analyse Zeitreihenkomponenten	T / LB	(Abbaszadeh Nakhost et al., 2025, S. 12–13)
9	Skalierbare, interoperable Infrastruktur	Deployment & Infrastructure	übergreifend	T / LB	(Dincelli & Jafarian, 2025, S. 325)

Nr.	Anforderung	Schicht	Prozessbezug	Herkunft	Quelle(n)
10	Robuste, vergleichende Evaluation gegen statistische Baselines statt Proof-of-Concept	Evaluation & Observability	Methodenwahl/ Prognoseerstellung/ Prognosefehler	LB	(Song et al., 2026, S. 13)
11	Laufendes Monitoring der KI-Leistung samt Störungsbewältigung und Notfallplänen	Evaluation & Observability	übergreifend	T / LB	(Hendriksen, 2023, S. 73)
12	Zuverlässigkeit und Fallback-Strategien	Evaluation & Observability	übergreifend	LN	(Xu et al., 2021, S. 14)
13	Kontrolle der Output-Stochastizität	Evaluation & Observability	Methodenwahl/ Prognoseerstellung/ Prognosefehler	LN	(Abbaszadeh Nakhost et al., 2025, S. 11–12)
14	Schutz vor Prompt Injection, Datenexfiltration und unautorisiertem Logging: Risiko der Memorisierung und Extraktion von Trainingsdaten	Security & Compliance	übergreifend	T / LB	(Carlini et al., 2021, S. 2640–2642)
15	Interoperabilität mit bestehenden Plaungssystemen und Supply Chain Informationssystemen	Agent Ecosystem	Integration Bedarfsplanung	LN	(Xu et al., 2021, S. 1, S. 14)
16	Human-in-the-Loop: keine volle Entscheidungsautonomie bei wirkungsstarken SCM-Entscheidungen, menschliche Genehmigung und Eingriffsmöglichkeit	Agent Ecosystem	Integration Bedarfsplanung	T / LB	(Jannelli et al., 2026, S. 18, S. 21; Trifone et al., 2026, S. 23–24)

Tabelle 4 - Liste technologischer Anforderungen an KI-Agenten im Forecasting von SCM

4.2 Ergebnisse FF2: Architektur der KI-Agenten

4.2.1 Vorgehen Darstellung Architektur

Die Gestaltung des Artefakts der Architektur der generativen KI-Agenten im Forecasting des SCM verbindet die beiden äußeren Zyklen des Modells mit dem Design Cycle. Aus dem Relevanz-Zyklus stammt der Anwendungskontext in Form des Forecasting-Prozesses in Anwendung auf das BPOS-Game, das festlegt, welche Daten verfügbar sind, welche Entscheidung gestützt werden soll und unter welchen Randbedingungen die Architektur operieren muss. Das Business Process Operations Simulation Game (BPOS-Game) der Plattform Education 4 Success ist ein netzwerkbasiertes Planspiel, das industrielle Geschäftsprozesse in einem Unternehmensnetzwerk in einer realitätsnahen Umgebung abbildet. Das Grundnetzwerk besteht aus vier miteinander verbundenen Unternehmen,

deren vertikale und horizontale Beziehungen die täglichen Aktivitäten und die finanzielle Leistung beeinflussen. Innerhalb jedes Teams werden vier Rollen entlang des „SCOR-Modells“ besetzt: Planner, Buyer, Operations und Distribution, sodass die Planungs-, Beschaffungs-, Produktions- und Distributionsfunktionen einer Lieferkette arbeitsteilig wahrgenommen werden. Das Spiel ist rundenbasiert und umfangreich parametrierbar: Produktnamen, Nachfragen, Qualitätsprobleme, Produktbaum, direkte und indirekte Kosten, Rundendauer sowie alternative Lieferanten und Kunden lassen sich konfigurieren, womit auch der Schwierigkeitsgrad steuerbar ist. Die Steuerung erfolgt datengetrieben: Eine fünfdimensionale Strategy Map mit zugehöriger Scorecard und Kennzahlen bildet den Bezugsrahmen für strategische Entscheidungen, die unter anderem Planung, Finanzen, Controlling und Data Science umfassen. Die im Spielverlauf erzeugten Daten werden in einer Datenbank vorgehalten, auf die alle Spielenden zugreifen und die sie mit gängigen Werkzeugen verbinden oder über vorkonfigurierte Dashboards auswerten können (Education 4 Success, o. J.). Diese Datenbasis bildet den Anknüpfungspunkt für die nachfolgend beschriebene Agenten-Konfiguration, die aus den historischen Nachfragezeitreihen Prognosen zur Stützung der Planungsentscheidung ableitet. Aus dem Rigor-Zyklus stammt der in der ersten Forschungsfrage erarbeitete Anforderungskatalog, dessen technologische Anforderungen die einzelnen Designentscheidungen begründen. Die Begründung der Architektur erfolgt in zwei Schritten. Zunächst wird hergeleitet, welche Agenten erforderlich sind: Die funktionale Gliederung folgt den Schritten des zuvor dargestellten Forecasting-Prozesses, sodass den Verarbeitungsschritten spezialisierte Agenten zugeordnet werden. Im zweiten Schritt wird die konkrete Ausgestaltung jedes Agenten beschrieben und anhand der einzelnen Anforderungen des Katalogs begründet und der jeweiligen Ebene des Sieben-Schichten-Modells zugeordnet.

4.2.2 CrewAI als Framework und Herleitung der Agenten

Das 2023 von João Moura veröffentlichte Python-Framework „CrewAI“ überträgt die Funktionsweise eines menschlichen Teams in Software. Das Resultat entsteht aus dem Zusammenspiel von Einzelbeiträge zu einem größeren Ganzen. Das Grundprinzip von CrewAI besteht darin, ein komplexes Problem durch ein orchestriertes Team rollenbasierter, LLM-gestützter Agenten zu lösen. Spezialisierte Agenten übernehmen dabei klar definierte Rollen, bearbeiten ihnen zugewiesene Aufgaben mithilfe geeigneter Werkzeuge, werden in einer Crew gebündelt, über einen Prozess koordiniert und durch ein gemeinsames Gedächtnis kontextuell verbunden. Ein einzelner Agent wird in CrewAI über eine Reihe von Konfigurationsfeldern definiert, die gemeinsam seine Identität, seinen Handlungsspielraum und sein Verhalten festlegen (Taulli & Deshmukh, 2025, S. 103–114). Nachfolgend werden die Anweisungen erklärt, die in der Konfiguration angewendet werden. Das Feld role bestimmt die fachliche Rolle des Agenten und legt damit fest, aus welcher Perspektive und mit welcher Kompetenz das Sprachmodell die ihm zugewiesene Aufgabe bearbeitet. Das Feld goal beschreibt das Ziel, auf das der Agent seine Handlungen ausrichtet. Das Feld backstory ergänzt die Rolle um einen fachlichen Hintergrund, der die Expertise und die Herangehensweise des Agenten prägt, Über diese identitätsstiftenden Felder hinaus steuern weitere Felder das konkrete Verhalten. Das Feld constraints definiert verbindliche Regeln, die den Handlungsspielraum des Agenten eingrenzen. Das Feld allow_delegation legt fest, ob der Agent Teilaufgaben an andere Agenten abgeben oder diese um Unterstützung bitten darf: mit dem Wert false ist dies unterbunden, sodass der Agent ausschließlich innerhalb seiner eigenen Aufgabe arbeitet. Das Feld verbose steuert die Ausführlichkeit der Protokollierung: mit dem Wert true werden die Verarbeitungsschritte und Werkzeugaufrufe des Agenten detailliert ausgegeben, was deren Nachvollziehbarkeit sicherstellt. Die beiden letztgenannten Felder sind für alle fünf Agenten der Architektur einheitlich gesetzt und stellen damit eine übergreifende Gestaltungsentscheidung dar. Die zugehörige Aufgabe wird analog über eigene Felder definiert. Das Feld description enthält die eigentliche Arbeitsanweisung und damit die fachliche Logik der Aufgabe. Das Feld expected_output legt die geforderte Struktur des Ergebnisses fest. Das Feld agent weist die Aufgabe dem ausführenden Agenten zu. Das Feld context weist jeder Aufgabe die Inhalte anderer

Aufgaben zu, da die Aufgaben die Informationen früherer Aufgaben benötigen, um abgearbeitet zu werden.

Die Herleitung der benötigten Agenten begründet sich an den Schritten des Forecasting-Prozesses und stellt sich wie folgt dar. Der erste Schritt, die Datenerfassung und -aufbereitung, verlangt die Bereitstellung vollständiger und konsistenter Nachfragezeitreihen als Grundlage jeder Prognose. Diese Funktion übernimmt der data_preparer, der die historischen Nachfragedaten beschafft, zusammenführt und in eine bereinigte Zeitreihe je Produkt überführt. Der zweite Schritt, die Analyse der Zeitreihenkomponenten, zerlegt die beobachtete Nachfrage in ihre systematischen Bestandteile und bildet damit die Voraussetzung für eine begründete Methodenwahl. Diese Aufgabe übernimmt der diagnostics_analyst, der jede Zeitreihe anhand erklärbarer Kennzahlen klassifiziert. Die folgenden drei Schritte: Methodenwahl, Prognoseerstellung und Bestimmung des Prognosefehlers, werden in einem einzigen Agenten, dem forecaster, zusammengeführt. Der Forecaster wählt je Zeitreihe das passende Modell aus einem definierten Modellpool, erstellt die Prognose und dokumentiert den zugehörigen Fehlerwert. Der sechste Schritt, die Integration in die Bedarfsplanung, überführt die erstellte Prognose in eine Form, die nachgelagerte SCM-Entscheidungen stützt. Dies übernimmt der management_advisor, der jede Prognose bewertet, den Nutzen offenlegt und Entscheidungsempfehlungen abgibt. Da die Prognose im vorliegenden Kontext nicht unmittelbar in eine autonome Entscheidung mündet, sondern einem menschlichen Entscheider zur Verfügung gestellt wird, übernimmt der data_reporter die Aufbereitung sämtlicher Ergebnisse zu einem strukturierten Bericht. Dadurch ergibt sich folgende Architektur aus fünf Agenten, welche zugleich das Artefakt dieser Arbeit darstellt.

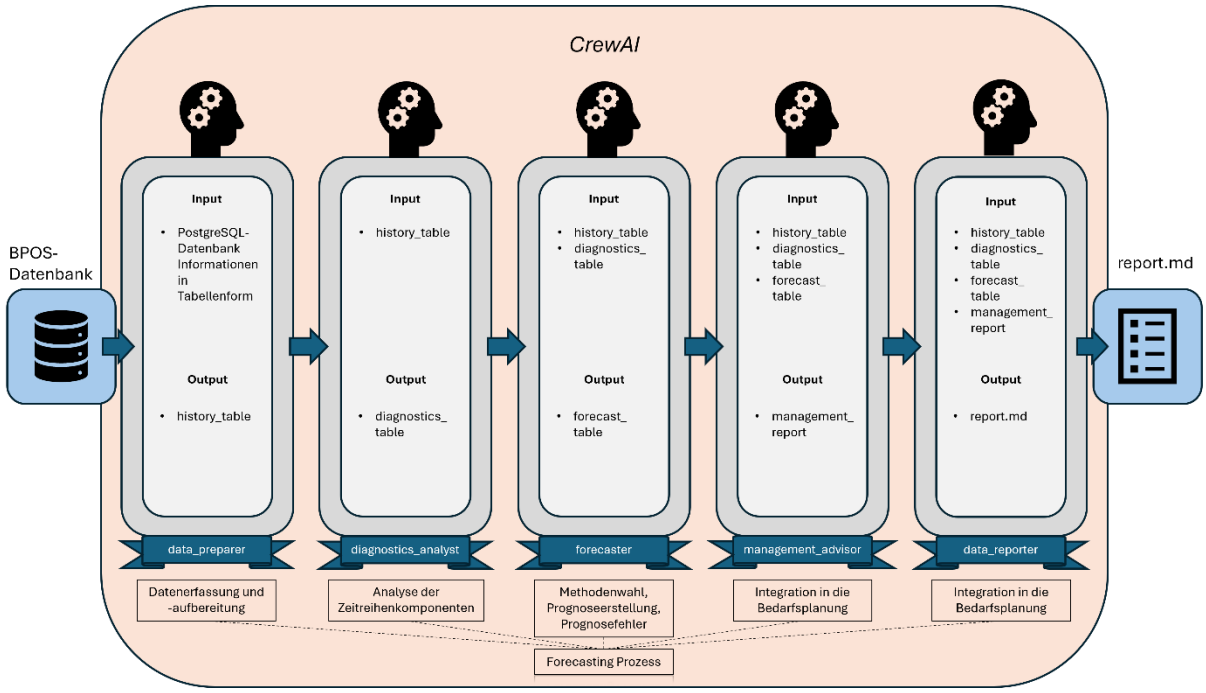


Abbildung 2 - Darstellung Artefakt

4.2.3 Konfiguration der Architektur in CrewAI

(1) Agent data_preparer mit Aufgabe prepare_history

```yaml
data_preparer:
  role: "Data Preparation & Historian"
  goal: >
    Erstelle saubere, lückenlose Nachfrage-Zeitreihen pro Produkt für die vier Produkte
    (prod13, prod14, prod15, prod16) basierend auf den Datenbankeinträgen.
    Liefere eine konsistente Tabelle für Forecasting.
  backstory: >
    Dateningenieur mit Fokus auf Zeitreihen, SQL-Joins und typische Datenqualitätsprobleme.
  constraints:
    - "Nutze ausschließlich das PostgreSQLReaderTool mit der definierten demand_history_query, um die Daten zu extrahieren."
    - "Achte darauf, dass jede Zeile der Tabellen orders und order_amounts gelesen und korrekt zusammengeführt wird."
    - "Achte auf besondere Genauigkeit beim Ablesen der Daten, um die Zeitreihen lückenlos und konsistent zu gestalten."
    - "Keine Schätzung oder Erfindung von Werten; nur die Daten verwenden, die tatsächlich in der Datenbank vorhanden sind."
  allow_delegation: false
  verbose: true
```

Abbildung 3 - Konfiguration des Agenten "data_preparer"

Im Fall des Agenten data_preparer ist die Rolle als „Data Preparation & Historian" definiert und weist ihn damit als zuständig für die Beschaffung und Historisierung der Nachfragedaten aus. Das Ziel ist dabei, saubere und lückenlose Nachfrage-Zeitreihen für die vier Produkte prod13 bis prod16 aus den Datenbankeinträgen zu erstellen und eine konsistente Tabelle als Grundlage für das Forecasting bereitzustellen. Der fachliche Hintergrund beschreibt den Agenten als Dateningenieur mit Schwerpunkt auf Zeitreihen, SQL-Joins und typischen Datenqualitätsproblemen. Die vier verbindlichen Regeln für den Agenten lauten: die ausschließliche Datenextraktion über das PostgreSQLReaderTool mit der definierten demand_history_query, das vollständige und korrekte Zusammenführen jeder Zeile der Tabellen orders und order_amounts, die besondere Sorgfalt zur Sicherung lückenloser und konsistenter Zeitreihen sowie das Verbot, Werte zu schätzen oder zu erfinden.

```yaml
prepare_history:
  description: >
    Extrahiere die Nachfragehistorie der vier Produkte prod13, prod14, prod15 und prod16
    ausschließlich über das PostgreSQLReaderTool.
    Verwende für die Weiterverarbeitung ausschließlich das vom Tool gelieferte vollständige
    history_table. Das Tool muss für alle aktuell in der Datenbank vorhandenen Runden
    (orders.round >= 0) pro Produkt genau einen Eintrag liefern.

    Die Tabelle muss folgende Felder enthalten:
      - product_id
      - round
      - demand_amount
      - status

    Fachliche Regeln:
      - Für jede aktuell vorhandene Runde muss für jedes der vier Produkte genau ein Eintrag existieren.
      - Falls für eine Produkt-Runden-Kombination kein Nachfragewert vorhanden ist, muss
        demand_amount = 0 gesetzt und status = Nullwert verwendet werden.
      - Falls für dieselbe Produkt-Runden-Kombination mehrere Einträge existieren, müssen diese
        summiert und mit status = aggregated gekennzeichnet werden.
      - Falls der aggregierte Nachfragewert negativ ist, muss status = invalid gesetzt werden.
      - Es dürfen keine Runden ausgelassen, verdichtet oder zusammengefasst werden.
      - Das history_table des Tools ist unverändert und vollständig zu übernehmen.

    Falls das Tool kein vollständiges history_table liefert, darf der Task nicht erfolgreich
    abgeschlossen werden.
  expected_output: >
    JSON mit:
      history_table: [{product_id, round, demand_amount, status}],
      round_count: int,
      product_count: int,
      expected_rows: int,
      actual_rows: int,
      nullwert_count: int
  agent: data_preparer
```

Abbildung 4 - Konfiguration der Aufgabe "prepare_history"

Die Aufgabe prepare_history umfasst die Extraktion der Nachfragehistorie der vier Produkte über das PostgreSQLReaderTool sowie die Vorgabe, für jede vorhandene Runde je Produkt genau einen Eintrag zu erzeugen. Hinzu treten feste fachliche Regeln zur Behandlung von Sonderfällen: Fehlende Werte werden als Nullwert gekennzeichnet, mehrfach vorhandene Einträge summiert und als aggregiert markiert, negative aggregierte Werte als ungültig ausgewiesen, und keine Runde darf ausgelassen oder verdichtet werden. Abschließend bestimmt die Anweisung, dass die Aufgabe nicht erfolgreich abgeschlossen werden darf, sofern keine vollständige Tabelle vorliegt. Das erwartete Ergebnis soll eine Tabelle der Nachfragezeitreihe (history_table mit den Feldern product_id, round, demand_amount und status) liefern und Prüfgrößen wie round_count, product_count, expected_rows, actual_rows und nullwert_count enthalten. Die Aufgabe wird dem data_preparer zugewiesen. Anders als die nachgelagerten Aufgaben verfügt prepare_history über kein context-Feld, da diese Aufgabe chronologisch an erster Stelle steht und somit die anderen Aufgaben auf ihr aufbauen.

Inhaltlich setzt die Konfiguration des data_preparer mehrere Anforderungen des Katalogs um und ist im Sieben-Schichten-Modell der Ebene der Data Operations zuzuordnen. Die Zielvorgabe lückenloser und konsistenter Zeitreihen sowie die fachlichen Regeln zur Behandlung fehlender, mehrfacher und ungültiger Werte setzen die Anforderung an die Datenqualität um, die inkonsistente oder fehlende Daten als unmittelbares Risiko für die Verlässlichkeit der Agenten-Outputs identifiziert (Anforderung 3). Dass die Datenaufbereitung als eigener Agent mit eigener Aufgabe von der Prognose getrennt ist, setzt die Anforderung um, Datenabruf und Vorverarbeitung als eigenständige Agentenfunktion auszugestalten (Anforderung 4). Der ausschließliche Datenbezug über das PostgreSQLRea-

derTool stellt eine Anbindung an die operative Datenbank des BPOS-Games in Echtzeit her und erfüllt die Anforderung an die Anbindung externer Datenquellen, allerdings nur in dem für den vorliegenden Kontext relevanten Umfang, ohne darüber hinausgehende externe Wissensquellen einzubinden (Anforderung 5). Das Verbot, Werte frei zu erzeugen, bindet die Datenbeschaffung an das Tool und folgt damit dem Prinzip des Tool-Groundings (Anforderung 6). Dass die Aufgabe nur als erfolgreich abgeschlossen werden darf, wenn eine vollständige Tabelle vorliegt, soll die Fortpflanzung eines Datenfehlers auf die nachgelagerten Agenten verhindern und dient damit der Zuverlässigkeit der Architektur (Anforderung 12). Die detaillierte Protokollierung über das Feld verbose und mit den Prüfgrößen im erwarteten Ergebnis soll die Nachvollziehbarkeit und Überwachung der Verarbeitung unterstützen und damit auch die Anforderung des laufenden Monitorings erfüllen (Anforderung 11).

(2) Agent diagnostics_analyst mit Aufgabe run_diagnostics

```yaml
diagnostics_analyst:
  role: "Demand Pattern & Change Analyst"
  goal: >
    Klassifiziere jede Produkt-Zeitreihe nach Intermittent, Trend und Saisonalität
    anhand erklärbarer Kennzahlen.
  backstory: >
    Supply-Chain-Analyst mit Fokus auf robuste Heuristiken.
  constraints:
    - "Klassifikationen nur aus nachvollziehbaren Kennzahlen ableiten und mitliefern."
    - "Bei unzureichenden Daten entsprechende Flags setzen und Hinweise geben."
  allow_delegation: false
  verbose: true
```

Abbildung 5 - Konfiguration des Agenten "diagnostics_analyst"

Im Fall des Agenten diagnostics_analyst ist die Rolle als „Demand Pattern & Change Analyst" definiert und weist ihn damit als zuständig für die Erkennung der Nachfragemuster in den aufbereiteten Nachfragezeitreihen aus. Das Ziel ist dabei, jede Produkt-Zeitreihe anhand erklärbarer Kennzahlen nach den Mustern Intermittent, Trend und Saisonalität zu klassifizieren. Der fachliche Hintergrund beschreibt den Agenten als Supply-Chain-Analyst mit Schwerpunkt auf robusten Heuristiken. Die zwei verbindlichen Regeln für den Agenten lauten: Klassifikationen ausschließlich aus nachvollziehbaren Kennzahlen abzuleiten und gemeinsam mit diesen auszuweisen sowie bei unzureichender Datengrundlage entsprechende Flags zu setzen und Hinweise zu geben.

```
run_diagnostics:
  description: >
    Analysiere ausschließlich das vollständige history_table aus prepare_history.
    Analysiere pro Produkt (prod13, prod14, prod15, prod16) die Nachfragehistorie history_table.
    Klassifiziere anhand erklärbarer Kennzahlen:
      - Intermittent
      - Trend
      - Saisonalität
    Falls keine Klassifizierung zutrifft, entsprechend kennzeichnen.
    Liefere alle Kennzahlen zur nachvollziehbaren Modellwahl.
  expected_output: >
    JSON mit:
      diagnostics_table: [{
        product_id,
        intermittent_flag,
        trend_flag,
        seasonality_flag,
        notes
      }],
      global_notes: string
  agent: diagnostics_analyst
  context: [prepare_history]
```

Abbildung 6 - Konfiguration der Aufgabe "run_diagnostics"

Die Aufgabe run_diagnostics umfasst die Analyse ausschließlich des vollständigen history_table aus der vorgelagerten Aufgabe prepare_history und ordnet jede Produkt-Zeitreihe anhand erklärbarer Kennzahlen den Mustern Intermittent, Trend und Saisonalität zu. Trifft keine Klassifizierung zu, ist dies entsprechend zu kennzeichnen. Ausdrücklich verlangt die Anweisung, alle Kennzahlen zu liefern, die eine nachvollziehbare Modellwahl ermöglichen. Das erwartete Ergebnis soll je Produkt eine Zeile in der diagnostics_table mit den Feldern product_id, intermittent_flag, trend_flag, seasonality_flag und notes sowie ein übergreifendes Feld global_notes liefern. Die Aufgabe wird dem diagnostics_analyst zugewiesen. Anders als prepare_history verfügt run_diagnostics über ein context-Feld, das auf prepare_history verweist und die Aufgabe damit an dessen Ergebnis bindet.

Inhaltlich setzt die Konfiguration des diagnostics_analyst den zweiten Schritt des Forecasting-Prozesses, die Analyse der Zeitreihenkomponenten, um und ist im Sieben-Schichten-Modell der Ebene der Agent Frameworks zuzuordnen. Dass die Diagnose als eigener Agent mit eigener Aufgabe von der Prognose getrennt ist, setzt die Anforderung um, die Zeitreihenmuster anhand erklärbarer Kennzahlen zu diagnostizieren und diese Diagnose zur Grundlage der Modellwahl zu machen (Anforderung 8). Die Regel, Klassifikationen nur aus nachvollziehbaren Kennzahlen abzuleiten und mitzuliefern, sichert zusätzlich die Erklärbarkeit der Diagnose ab. Die Vorgabe, bei unzureichender Datengrundlage Flags zu setzen, soll unsichere Klassifikationen ausweisen, statt zu verdecken und dient damit der Zuverlässigkeit der Architektur (Anforderung 12). Die Bindung der Aufgabe über das context-Feld an das Ergebnis von prepare_history stellt sicher, dass die Diagnose auf einer qualitätsgesicherten Datengrundlage aufsetzt, und verkettet damit die Datenqualität des ersten Schrittes mit der Mustererkennung des zweiten (Anforderung 3). Die detaillierte Protokollierung über das Feld verbose sowie die erläuternden Felder notes und global_notes im erwarteten Ergebnis sollen die Nachvollziehbarkeit und Überwachung der Klassifikation unterstützen und damit auch die Anforderung des laufenden Monitorings erfüllen (Anforderung 11).

(3) Agent: forecaster mit Aufgabe: create_forecast

```yaml
forecaster:
  role: "Forecasting Specialist"
  goal: >
    Wähle je Zeitreihe der Produkte die beste Prognose aus dem definierten Modellpool
    (Naive, SES, Holt, Holt-Winters, SBA). Und berechne die Prognose je Produkt für die nächsten beiden Perioden (h=1 und h=2).
  backstory: >
    Forecasting Engineer mit Fokus auf robuste, datenarme Modelle.
  constraints:
    - "Nur Modelle aus dem Pool verwenden und model_used dokumentieren."
    - "Forecasts nur aus Tool-Runs; Fallback auf Naive/SES mit Begründung."
    - "Keine Freitext-Berechnung von Prognosewerten."
  allow_delegation: false
  verbose: true
```

Abbildung 7 - Konfiguration des Agenten "forecaster"

Im Fall des Agenten forecaster ist die Rolle als „Forecasting Specialist" definiert und weist ihn damit als zuständig für die Auswahl und Berechnung der Prognosen aus. Das Ziel ist dabei, je Produkt-Zeitreihe die beste Prognose aus dem definierten Modellpool Naive Prognose, Einfache Exponentielle Glättung (SES), Holts Modell, Winters-Modell und SBA auszuwählen und die Prognose für die nächsten beiden Perioden (h=1 und h=2) zu berechnen. Die Beschränkung auf die fünf Forecasting-Methoden ist eine bewusste Designentscheidung. Zum einen werden für die benchmark-basierte Evaluation nachvollziehbare Referenzprognosen benötigt, die sich reproduzierbar nachrechnen lassen. Zum anderen deckt der gewählte Pool die im diagnostics_analyst unterschiedenen Zeitreihencharakteristika vollständig ab: SES erfasst ein reines Niveau, Holts Modell zusätzlich einen Trend, Winters Modell Trend und Saisonalität und die SBA intermittierende Nachfrage, während die Naive Prognose als Basismodell und Fallback dient. Damit steht für jedes diagnostizierbare Muster ein passendes Verfahren bereit, sodass die Methodenwahl unmittelbar an die Diagnose anschließt und die Verfahren zugleich als Referenzmethoden der Evaluation fungieren. Der Prognosehorizont von zwei Perioden ergibt sich aus der Position als Tier-2-Akteur im Liefernetzwerk des BPOS-Games. Da Bestellung und Produktion zusammen eine Durchlaufzeit von zwei Runden beanspruchen, müssen die heute zu treffenden Beschaffungs- und Produktionsentscheidungen die Nachfrage über genau dieses Zeitfenster antizipieren. Die Prognose für die nächsten beiden Perioden (h=1 und h=2) bildet damit exakt den Zeitraum ab, über den aktuelle Entscheidungen wirksam werden. Der fachliche Hintergrund beschreibt den Agenten als Forecasting Engineer mit Schwerpunkt auf robusten, datenarmen Modellen. Die drei verbindlichen Regeln für den Agenten lauten: ausschließlich Modelle aus dem Pool zu verwenden und das verwendete Modell zu dokumentieren, Prognosen nur aus Tool-Runs zu erzeugen und einen Fallback auf Naive oder SES zu begründen sowie keine Prognosewerte im Freitext zu berechnen.

```yaml
create_forecast:
  description: >
    Erstelle je Produkt (prod13, prod14, prod15, prod16) Prognosen für die nächsten beiden Perioden (h=1 und h=2) auf Basis von
    history_table und diagnostics_table. Benutze dabei je nach Diagnostik der Zeitreihe das passende Prognosemodell
    aus den Modellen: Naive Prognose, Einfache Exponentielle Glättung (nur wenn Anzahl Perioden > 3), Holt-Methode (nur wenn Trend vorhanden),
    Holt-Winters-Methode (nur Trend und Saisonalität vorhanden),
    Syntetos-Boylan-Approximation (nur bei intermittierender Nachfrage).
    Wähle datengetrieben (z.B. Rolling Backtest); bei Fehlschlag oder
    zu wenig Daten fallback auf Naive oder SES und dokumentiere den Grund.
  expected_output: >
    JSON mit:
      forecast_table: [{
        product_id,
        forecast_time_index,
        horizon,  # nur Werte 1 oder 2
        yhat,
        model_used,
        model_params (OPTIMIERT),
        optimization_performed,
        selection_method,
        backtest_score,
        notes
      }],
      Erwartung: genau 2 Einträge pro Produkt (horizon=1 und horizon=2).
      model_selection_summary: string
  agent: forecaster
  context: [prepare_history, run_diagnostics]
```

Abbildung 8 - Konfiguration der Aufgabe "create_forecast"

Die Aufgabe create_forecast umfasst die Erstellung der Prognosen je Produkt für die nächsten beiden Perioden auf Basis von history_table und diagnostics_table. Je nach diagnostiziertem Muster ist das passende Modell zu wählen, wobei feste Anwendungsvoraussetzungen gelten: SES ist nur bei mehr als drei Perioden zulässig, Holts Modell nur bei vorhandenem Trend, Winters Modell nur bei Trend und Saisonalität und SBA nur bei intermittierender Nachfrage. Die Auswahl erfolgt datengetrieben, etwa über ein Rolling Backtest; schlägt sie fehl oder liegen zu wenige Daten vor, ist auf die Naive Prognose oder SES zurückzufallen und der Grund zu dokumentieren. Das erwartete Ergebnis soll je Produkt genau zwei Einträge (für die nächsten beiden Perioden) in der forecast_table liefern, mit Feldern wie product_id, yhat, model_used, model_params, optimization_performed, selection_method, backtest_score und notes sowie eine übergreifende model_selection_summary. Die Aufgabe wird dem forecaster zugewiesen. Über das context-Feld ist sie sowohl an prepare_history als auch an run_diagnostics gebunden und greift damit zugleich auf die bereinigte Zeitreihe und die zugehörige Diagnose zu.

Inhaltlich fasst die Konfiguration des forecaster drei Schritte des Forecasting-Prozesses zusammen: Methodenwahl, Prognoseerstellung und Bestimmung des Prognosefehlers. Die Konfiguration ist im Sieben-Schichten-Modell der Ebene der Agent Frameworks zuzuordnen. Die Anforderung des Tool-Groundings wird hierbei nur zum Teil erfüllt. Prognosen sollen zwar über vorgegebene Methoden berechnet werden, allerdings erfolgt dies nicht über externe Tools. Der Grund hierfür liegt in der späteren Bewertung hinsichtlich der Evaluation: Inwieweit können generative KI-Agenten im Forecasting des SCM nützlich sein. Dies wird später in den Limitationen aufgegriffen. Die datengetriebene Modellauswahl über ein Rolling Backtest samt dokumentiertem backtest_score setzt die Anforderung um, die Prognosen vergleichend gegen die Modellalternativen zu bewerten (Anforderung 10). Außerdem wird zugleich die Bestimmung des Prognosefehlers als integraler Bestandteil der Modellauswahl operationalisiert. Die Fallback-Regel auf Naive oder SES mit dokumentierter Begründung sichert den Prognoseschritt gegen Fehlschläge und unzureichende Daten ab und dient der Zuverlässigkeit der Architektur (Anforderung 12). Die Bindung der Aufgabe über das context-Feld an run_diagnostics stellt sicher, dass die Modellwahl auf der zuvor erstellten Diagnose aufsetzt, und setzt damit die Anforderung um, die Diagnose der Zeitreihenmuster zur Grundlage der Modellwahl zu machen (Anforderung 8). Die Protokollierung über das Feld verbose sowie die Dokumentation von model_used, selection_method und backtest_score im erwarteten Ergebnis sollen die Nachvollziehbarkeit und Überwachung der Prognoseerstellung unterstützen und damit auch die Anforderung des laufenden Monitorings erfüllen (Anforderung 11).

(4) Agent: management_advisor mit Aufgabe: evaluate_management

```yaml
management_advisor:
  role: "Management- und Prognosebewertungs-Agent"
  goal: >
    Bewerte die Ergebnisse der bisherigen Nachfrage- und Prognoseagenten kritisch aus Managementsicht,
    leite konkrete Handlungsempfehlungen ab und beurteile die methodische Angemessenheit sowie die
    praktische Nutzbarkeit der Prognosen.
  backstory: >
    Erfahrener Experte für Nachfrageplanung, Bestandsmanagement und Prognosebewertung.
  constraints:
    - "Bewertung der Ergebnisse aus der Perspektive eines erfahrenen Management-Experten."
    - "Konkrete, umsetzbare Handlungsempfehlungen ableiten."
    - "Kritische Beurteilung der methodischen Angemessenheit und praktischen Nutzbarkeit der Prognosen."
  allow_delegation: false
  verbose: true
```

Abbildung 9 - Konfiguration des Agenten "management_advisor"

Im Fall des Agenten management_advisor ist die Rolle als „Management- und Prognosebewertungs-Agent" definiert und weist ihn damit als zuständig für die kritische Bewertung der bisherigen Ergebnisse aus Managementsicht aus. Das Ziel ist dabei, die Ergebnisse der vorgelagerten Nachfrage- und

Prognoseagenten kritisch zu bewerten, konkrete Handlungsempfehlungen abzuleiten und sowohl die methodische Angemessenheit als auch die praktische Nutzbarkeit der Prognosen zu beurteilen. Der fachliche Hintergrund beschreibt den Agenten als erfahrenen Experten für Nachfrageplanung, Bestandsmanagement und Prognosebewertung. Die drei verbindlichen Regeln für den Agenten lauten: die Ergebnisse aus der Perspektive eines erfahrenen Management-Experten zu bewerten, konkrete und umsetzbare Handlungsempfehlungen abzuleiten sowie die methodische Angemessenheit und praktische Nutzbarkeit der Prognosen kritisch zu beurteilen.

```yaml
evaluate_management:
  description: >
      Analysiere die Ergebnisse der vorherigen Agenten: historische Nachfragewerte, diagnostizierte
      Nachfragemuster und Prognosen für die nächsten zwei Perioden. Erstelle daraus eine strukturierte
      managementorientierte Bewertung.
      Für jedes Produkt sollst du:
      - den Nutzen der Prognose für das Management beschreiben
      - konkrete Handlungsempfehlungen geben
      - die Eignung des gewählten Prognoseverfahrens bewerten
      - Risiken, Unsicherheiten und Einschränkungen benennen
      - eine Einschätzung zur Belastbarkeit der Prognose geben
      Erstelle anschließend ein Gesamturteil darüber, ob das bisherige Vorgehen fachlich sinnvoll war,
      wo Schwächen liegen und welche Verbesserungen künftig empfohlen werden.
  expected_output: >
      Eine strukturierte Bewertung "management_report" mit: Management-Zusammenfassung, produktbezogenen Empfehlungen,
      kritischer Methodenbewertung, Risikoanalyse, Gesamturteil zum Vorgehen und Verbesserungsvorschlägen.
  agent: management_advisor
  context: [prepare_history, run_diagnostics, create_forecast]
```

Abbildung 10 - Konfiguration der Aufgabe "evaluate_management"

Die Aufgabe evaluate_management umfasst die Analyse der Ergebnisse der vorherigen Agenten und deren Verdichtung zu einer strukturierten, managementorientierten Bewertung. Je Produkt sind dabei der Nutzen der Prognose für das Management zu beschreiben, konkrete Handlungsempfehlungen zu geben, die Eignung des gewählten Prognoseverfahrens zu bewerten, Risiken, Unsicherheiten und Einschränkungen zu benennen sowie eine Einschätzung zur Belastbarkeit der Prognose abzugeben. Abschließend ist ein Gesamturteil darüber zu erstellen, ob das bisherige Vorgehen fachlich sinnvoll war, wo Schwächen liegen und welche Verbesserungen künftig empfohlen werden. Das erwartete Ergebnis ist eine strukturierte Bewertung in Form des management_report mit Management-Zusammenfassung, produktbezogenen Empfehlungen, kritischer Methodenbewertung, Risikoanalyse, Gesamturteil zum Vorgehen und Verbesserungsvorschlägen. Die Aufgabe wird dem management_advisor zugewiesen. Über das Feld context ist sie an prepare_history, run_diagnostics und create_forecast gebunden und greift damit erstmals auf die Ergebnisse aller drei vorgelagerten Agenten zu, was ihrer Rolle als bewertende Instanz über die gesamte Verarbeitungskette entspricht.

Der management_advisor setzt an der Integration des Forecasts in die Bedarfsplanung und in das SCM an. Iim Sieben-Schichten-Modell ist er vorrangig der Ebene des Agent Ecosystem zuzuordnen, mit Bezügen zur Ebene Evaluation & Observability. Konzeptionell greift er die Agentenkomponente der Reflexion auf, also die Fähigkeit, die vorgelagerten Verarbeitungsergebnisse zu prüfen und einzuordnen. Die kritische Bewertung der Prognosen und die Beurteilung der methodischen Angemessenheit setzen die Anforderung um, die Prognosen kritisch und vergleichend zu bewerten, statt sie ungeprüft zu übernehmen (Anforderung 10). Dass der Agent nicht selbst eine Bestellentscheidung trifft, sondern Handlungsempfehlungen und eine managementorientierte Bewertung als Entscheidungsgrundlage für den Menschen erzeugt, versucht die Anforderung an den Human-in-the-Loop umzusetzen, damit SCM-Entscheidungen nicht vollständig automatisiert ablaufen, sondern dem Menschen Genehmigung und Eingriff vorbehalten (Anforderung 16). Die ausdrückliche Benennung von Risiken, Unsicherheiten und Einschränkungen sowie die Einschätzung zur Belastbarkeit der Prognosen stützen dieses Prinzip zusätzlich, da sie dem menschlichen Entscheider die für eine fundierte Entscheidung nötigen Informationen bereitstellen. Auch hier soll die ausführliche Protokollierung über das Feld verbose die Nachvollziehbarkeit der Bewertung unterstützen (Anforderung 11).

(5) Agent: data_reporter mit Aufgabe: report_data

```yaml
data_reporter:
  role: "Data & Forecasting Report Generator"
  goal: >
    Erstelle einen strukturierten Bericht über history_table, diagnostics_table, forecast_table und management_report
    mit klaren Überschriften und Absätzen.
    Alle Einträge der Tabellen vollständig darstellen.
  backstory: >
    Reporting Engineer für Analyse- und Forecast-Ergebnisse.
  constraints:
    - "Ergebnisse klar und übersichtlich darstellen."
    - "Keine neuen Werte erfinden; nur vorhandene Ergebnisse darstellen."
    - "Alle Ergebnisse darstellen und kein weglassen von Informationen."
  allow_delegation: false
  verbose: true
```

Abbildung 11 - Konfiguration des Agenten "data_reporter"

Im Fall des Agenten data_reporter ist die Rolle als „Data & Forecasting Report Generator" definiert und weist ihn damit als zuständig für die strukturierte Aufbereitung und Darstellung der Ergebnisse aus. Das Ziel ist dabei, einen strukturierten Bericht über history_table, diagnostics_table, forecast_table und mangement_report mit klaren Überschriften und Absätzen zu erstellen und sämtliche Einträge der Tabellen vollständig darzustellen. Der fachliche Hintergrund beschreibt den Agenten als Reporting Engineer für Analyse- und Forecast-Ergebnisse. Die drei verbindlichen Regeln für den Agenten lauten: die Ergebnisse klar und übersichtlich darzustellen, keine neuen Werte zu erfinden, sondern ausschließlich vorhandene Ergebnisse wiederzugeben sowie alle Ergebnisse darzustellen und keine Informationen auszulassen.

```yaml
report_data:
  description: >
    Erstelle einen Markdown-Bericht aus den vorherigen JSON-Ergebnissen mit:
      - history_table mit allen Einträgen
      - diagnostics_table mit allen Einträgen
      - forecast_table mit allen Einträgen
      - OPTIMIERTE Glättungsparameter (alpha, beta, gamma) mit Optimierungsergebnis
      - Backtest MSE Scores
      - management_report mit allen Einträgen
    Klare Überschriften und Absätze.
    Dokumentiere, welche Parameter AUTOMATISCH optimiert wurden (optimization_performed=true).
  expected_output: >
    Markdown-Datei 'report.md'
  agent: data_reporter
  context: [prepare_history, run_diagnostics, create_forecast, evaluate_management]
```

Abbildung 12 - Konfiguration der Aufgabe "report_data"

Die Aufgabe report_data umfasst die Erstellung eines Markdown-Berichts aus den JSON-Ergebnissen der vorherigen Agenten. Der Bericht enthält die vollständigen Einträge der history_table, der diagnostics_table und der forecast_table, die optimierten Glättungsparameter (alpha, beta, gamma) samt Optimierungsergebnis, die Backtest-MSE-Werte sowie den vollständigen management_report. Ergänzend ist zu dokumentieren, welche Parameter automatisch optimiert wurden. Das erwartete Ergebnis ist eine Markdown-Datei mit dem Namen report.md, die beim Lauf der Crew als Ausgabedatei festgelegt ist. Die Aufgabe wird dem data_reporter zugewiesen. Über das context-Feld ist sie an prepare_history, run_diagnostics, create_forecast und evaluate_management gebunden und greift damit als letzte Aufgabe der Kette auf die Ergebnisse aller vier vorgelagerten Agenten zu, die sie zu einem zusammenhängenden Bericht verdichtet.

Der data_reporter bildet den Abschluss der Verarbeitungskette und überführt die Ergebnisse aller Agenten in eine strukturierte, für den Menschen lesbare Form und unterstützt dadurch die Integration des Forecasts in die Bedarfsplanung des SCM. Im Sieben-Schichten-Modell ist er der Ebene des

Agent Ecosystem zuzuordnen, mit Bezügen zur Ebene Evaluation & Observability. Dass der erzeugte Bericht die Entscheidungsgrundlage für den menschlichen Entscheider bildet und nicht selbst in eine automatisierte Entscheidung mündet, setzt die Anforderung an den Human-in-the-Loop um, wonach wirkungsstarke SCM-Entscheidungen dem Menschen zur Genehmigung und zum Eingriff vorbehalten bleiben (Anforderung 16). Die Regeln, keine neuen Werte zu erfinden und sämtliche Ergebnisse vollständig sowie ohne Auslassung darzustellen, sichern die Integrität und Vollständigkeit der Berichterstattung und stellen sicher, dass der Bericht die Ergebnisse der vorgelagerten Agenten unverfälscht wiedergibt. Gemeinsam mit der Protokollierung über das Feld verbose unterstützt dies die Nachvollziehbarkeit und Überwachung der Verarbeitung (Anforderung 11).

(6) Übergreifende Architektur

Der folgende Überblick zeigt und erläutert die zentralen Dateien des Projekts und ihre jeweilige Funktion, um den Aufbau des Artefakts nachvollziehbar zu machen.

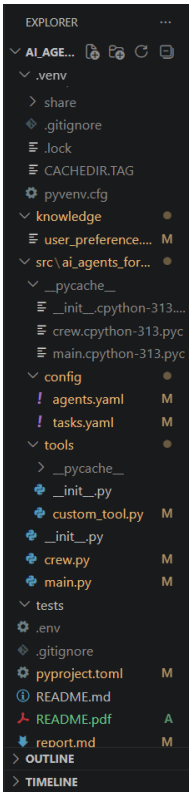


Abbildung 13 - Darstellung weiterer relevanter Dateien der übergreifenden Konfiguration

„crew.py“ bildet das Rückgrat der Architektur: Sie instanziiert die fünf Agenten und die fünf Aufgaben aus den Konfigurationsdateien, weist die Werkzeuge zu und bündelt alles zu einer Crew, deren Aufgaben über einen sequenziellen Prozess nacheinander ausgeführt werden. „agents.yaml“ enthält die Konfiguration der fünf Agenten und „tasks.yaml“ enthält die Konfiguration der fünf Aufgaben. „custom_tool.py“ definiert das deterministische Tool: das „PostgreSQLReaderTool“, ein Tool zur Datenextraktion, was mittels Verknüpfung der relevanten Tabellen in der Datenbank die relevanten Informationen für den „data_preparer“ bereitstellt. „main.py“ bildet den Einstiegspunkt der Anwendung: Sie lädt die Umgebungsvariablen, startet die Crew und erzeugt aus den Ergebnissen den Bericht (report.md sowie eine PDF-Fassung). Zusätzlich protokolliert sie den Tokenverbrauch je Durchlauf. „.env“ hinterlegt die Umgebungsvariablen, insbesondere das verwendete Sprachmodell, den API-Schlüssel und die Zugangsdaten zur Datenbank, und wird nicht versioniert. „.gitignore“ legt fest, welche Dateien von der Versionsverwaltung ausgeschlossen werden, darunter die „.env“ mit den sensiblen Zugangsdaten. „pyproject.toml“ beschreibt das Projekt als eigenständiges Python-Paket

und verwaltet dessen Abhängigkeiten. „knowledge/“ ist das von CrewAI vorgesehene Verzeichnis für Wissensquellen und hält hier hinterlegte Nutzerpräferenzen vor, auf die die Crew zugreifen kann. „tests/“ ist für Testskripte zur Überprüfung der Funktionsfähigkeit vorgesehen. „README.md“ (sowie die PDF-Fassung „README.pdf“) enthält eine standardisierte Projektdokumentation mit Hinweisen zu Aufbau und Nutzung. „report.md“ ist die vom System erzeugte Ausgabedatei mit den zusammengeführten Ergebnistabellen und der Managementbewertung. „.venv“ ist die isolierte virtuelle Umgebung, die die installierten Abhängigkeiten des Projekts enthält, und „__init__.py“ kennzeichnet die Verzeichnisse als Python-Pakete (das automatisch erzeugte „__pycache__“ enthält lediglich kompilierten Bytecode).

Auch hier werden mehrere Anforderungen auf der Ebene der übergreifenden Projektstruktur und des unterstützenden Programmcodes umgesetzt. Das zugrunde liegende Sprachmodell wird nicht im Agentencode, sondern über eine Umgebungsvariable konfiguriert: Die Datei .env legt mit dem Eintrag MODEL das verwendete Foundation Model fest. Im hier vorliegenden Fall das Modell GPT-4.1-mini von OpenAI und stellt über den OPENAI_API_KEY die Authentifizierung bereit. Diese Variablen werden beim Programmstart geladen. Damit wird die Anforderung an ein leistungsfähiges Sprachmodell als kognitive Basis umgesetzt, das im Sieben-Schichten-Modell der Ebene der Foundation Models zuzuordnen ist (Anforderung 1). Die Beobachtbarkeit der Verarbeitung wird über mehrere Mechanismen sichergestellt. Neben der detaillierten Protokollierung auf Agenten- und Crew-Ebene (verbose) erfasst der Programmcode in main.py je Durchlauf den Tokenverbrauch und legt ihn zusammen mit einem Zeitstempel in einer eigenen Datei ab. Jeder Durchlauf wird zudem in einem fortlaufend nummerierten Verzeichnis dokumentiert. Damit wird ein weiteres Mal versucht, die Anforderung an ein laufendes Monitoring der Verarbeitung im Rahmen des Prototyps umzusetzen und der Ebene Evaluation & Observability zugeordnet (Anforderung 11). Die Datenanbindung erfolgt über eine psycopg2-Verbindung an die operative PostgreSQL-Datenbank des BPOS-Games, während die Ergebnisse über standardisierte JSON- und CSV-Strukturen ausgetauscht werden. Darin wird versucht, die Interoperabilität mit dem operativen Informationssystem sicherzustellen (Anforderung 15).

Auf der Ebene der Infrastruktur ist das Projekt als eigenständiges Python-Paket mit einer src-Struktur organisiert, nutzt eine isolierte virtuelle Umgebung und trennt Konfiguration, Werkzeuge und Orchestrierung modular voneinander. Dies stellt eine reproduzierbare, jedoch lokal betriebene Ausführungsumgebung bereit und setzt die Anforderung an eine skalierbare Infrastruktur damit nur teilweise um (Anforderung 9). Die sicherheitsrelevante Behandlung der Zugangsdaten erfolgt durch deren Auslagerung in die .env-Datei, die über die .gitignore von der Versionierung ausgeschlossen ist, sowie durch eine Validierung der erforderlichen Verbindungsparameter im PostgreSQLReaderTool. Eine darüber hinausgehende Absicherung gegen weitergehende Bedrohungen findet jedoch nicht statt, sodass auch die Anforderung: Schutz vor Prompt-Injection oder Datenextraktion nur teilweise erfüllt wird (Anforderung 14).

4.2.4 Umgesetzte Anforderungsliste

Anhand der Konfiguration einzelner Agenten und Aufgaben sowie der übergreifenden Architektur ergibt sich nun folgende Umsetzung, der in Forschungsfrage 1 aufgestellten Liste an technologischen Anforderungen an KI-Agenten im Forecasting von SCM:

Nr.	Anforderung	Schicht	Umsetzung im Artefakt	Erfüllungsgrad
1	Leistungsfähiges LLM als kognitive Basis	Foundation Models	Übergreifend: Modellkonfiguration in .env (Eintrag MODEL), Authentifizierung über OPENAI_API_KEY.	vollständig
2	Strukturierte Schritt-für-Schritt-Prompts	Foundation Models	Übergreifend: strukturierte task-descriptions mit nummerierten fachlichen Regeln und festen expected_output-Schemata.	vollständig
3	Datenqualität: vollständige, konsistente Nachfragezeitreihen	Data Operations	data_preparer / prepare_history	vollständig
4	Datenabruf und Vorverarbeitung als eigenständige Agentenfunktionen	Data Operations	data_preparer als eigener Agent mit eigener Aufgabe, von der Prognose getrennt.	vollständig
5	Echtzeit-Anbindung externer Daten- und Wissensquellen	Data Operations	data_preparer: PostgreSQLReaderTool an die operative BPOS-Datenbank, keine darüber hinausgehenden externen Wissensquellen.	teilweise
6	Tool-Grounding	Agent Frameworks	data_preparer: ausschließlich PostgreSQLReaderTool zur Verknüpfung von Tabellen in Datenbank, aber: forecaster macht Prognoseberechnung nicht über externe Tools.	teilweise
7	Mathematische Passung des Tools	Agent Frameworks	PostgreSQLReaderTool verknüpft Tabellen erfolgreich, aber: forecaster nutzt Modelle nur anhand von Prompts und nicht anhand von Tools.	teilweise
8	Erklärbare Diagnose der Zeitreihenmuster als Grundlage der Modellwahl	Agent Frameworks	diagnostics_analyst / run_diagnostics: Klassifikation mit Kennzahlen und notes. forecaster nutzt Diagnose über context.	vollständig
9	Skalierbare, interoperable Infrastruktur	Deployment & Infrastructure	Übergreifend: src-Paketstruktur, isolierte virtuelle Umgebung, modu-	teilweise

Nr.	Anforderung	Schicht	Umsetzung im Artefakt	Erfüllungsgrad
			lare Trennung von Konfiguration/Tools/Orchestrierung, aber alles lokal betrieben.	
10	Robuste, vergleichende Evaluation gegen statistische Baselines statt Proof-of-Concept	Evaluation & Observability	forecaster: Rolling Backtest + backtest_score, management_advisor: kritische Methodenbewertung. Es wird erst vollständig über das FF3-Evaluationsdesign.	teilweise (Artefakt)
11	Laufendes Monitoring der KI-Leistung samt Störungsbewältigung und Notfallplänen	Evaluation & Observability	Übergreifend: Token-Tracking je Lauf (main.py), nummerierte Run-Verzeichnisse, verbose-Protokollierung auf Agenten- und Crew-Ebene.	teilweise
12	Zuverlässigkeit und Fallback-Strategien	Evaluation & Observability	data_preparer (Bedingung zur erfolgreichen Abschließung der Aufgabe), diagnostics_analyst (Flags bei unzureichenden Daten), forecaster (Fallback auf Naive/SES mit Begründung).	vollständig
13	Kontrolle der Output-Stochastizität	Evaluation & Observability	Im Artefakt wurden zwar die Reports ausgegeben und anhand der Iterationsschritte evaluiert und verbessert, allerdings passiert die ausführliche Evaluation in FF3.	teilweise (Artefakt)
14	Schutz vor Prompt Injection, Datenexfiltration und unautorisiertem Logging	Security & Compliance	Übergreifend: Credentials in .env ausgelagert und per .gitignore von der Versionierung ausgeschlossen, Validierung der Verbindungsparameter im PostgreSQLReaderTool. Aber: kein Injection-Schutz.	teilweise
15	Interoperabilität mit bestehenden ERP- und SC-Informationssystemen	Agent Ecosystem	Übergreifend / data_preparer: psycopg2-Anbindung an die BPOS-Datenbank, Nutzung von JSON-/CSV-Schnittstellen. Aber: keine reale ERP-Integration.	teilweise
16	Human-in-the-Loop: keine volle Entscheidungsautonomie,	Agent Ecosystem	management_advisor (Bewertung + Handlungsempfehlungen statt Entscheidung), data_reporter (Bericht als Entscheidungsgrundlage).	vollständig

Nr.	Anforderung	Schicht	Umsetzung im Artefakt	Erfüllungsgrad
	menschliche Genehmigung und Eingriffsmöglichkeit			

Tabelle 5 - Darstellung umgesetzter Anforderungen aus der Tabelle 4

Auffallend hierbei ist, dass versucht wurde alle Anforderungen umzusetzen. Jedoch wurden die meisten Anforderungen nur teilweise umgesetzt, was Themen für zukünftige Forschung liefert und im entsprechenden Kapitel beschrieben wird.

4.2.5 Iterative Verfeinerung des Artefakts

Die Konstruktion der Multi-Agenten-Architektur folgte der Build-and-Evaluate-Schleife des Design Cycle, die über wiederholte Durchläufe aus Konstruktion, Evaluation und Rückkopplung den Entwurf verfeinert (Hevner, 2007, S. 88, S. 90; Hevner et al., 2004, S. 78). Maßgeblich waren dabei die in Probeläufen beobachteten Defizite, die als formative, entwicklungsbegleitende Evaluation die jeweils folgende Überarbeitung des Artefakts begründeten. Diese Bewertung ist von der Evaluation der fertigen Architektur (FF3) zu unterscheiden. Im Folgenden werden zwei Korrekturebenen exemplarisch dargestellt.

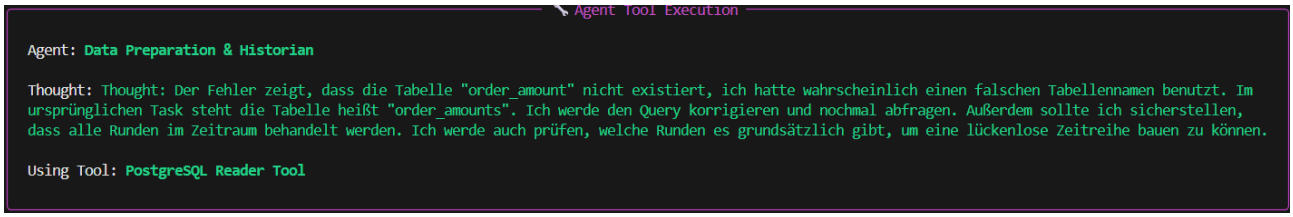


Abbildung 14 - Beispiel Fehler im Arbeitsablauf der Agenten

Abbildung 14 zeigt, wie der data_preparer einen fehlerhaften Tabellenbezug erkennt: die abgefragte Tabelle „order_amount" existiert nicht in der Datenbank, korrekt ist „order_amounts". Daraufhin korrigierte er die Abfrage eigenständig und führte diese über das PostgreSQLReaderTool erneut aus. Sobald ein solcher Fehler während des Durchlaufs der KI-Agenten erkannt wurde, wurde dieser behoben. Außerdem führten beobachtete Fehler in den erzeugten Reports zu einer Überarbeitung des Artefakts durch den Entwickler.

```
description: >
  Extrahiere Nachfragehistorie durch JOIN:
  SELECT o.round, oa.product_id, oa.amount
  FROM orders o
  INNER JOIN order_amounts oa ON o.order_id = oa.order_id
  WHERE oa.product_id IN ('prod17', 'prod18', 'prod19', 'prod20')
    AND o.round >= 0
  ORDER BY oa.product_id, o.round

expected_output: >
  JSON mit:
    - history_table: [{product_id, round_index, demand_per_round, status}]
    - last_round_index
agent: data_preparer
```

Abbildung 15 - Auszug aus früherer Konfiguration der Aufgabe "prepare_history"

In einer frühen Iteration war die Datenextraktion als SQL-Abfrage unmittelbar in den data_preparer-Task eingebettet, wie in Abbildung 15 dargestellt.

```
## 1. Datenqualität und Nachfragehistorie

Die Datenqualität der vorliegenden Nachfragehistorien für alle Produkte (`prod17`
bis `prod20`) ist durchgehend als **"ok"** bewertet, ohne fehlende Werte oder
Nullnachfrage. Dies ermöglicht eine stabile Analyse und Prognose.

Im Folgenden werden die historischen Nachfragedaten je Produkt in tabellarischer
Form aufgeführt:

| product_id | round_index | amount |
|------------|-------------|--------|
<!-- Beginn Daten zeilen für alle Produkte -->

<!-- Prod17 Nachfragehistorie: 240 Runden -->
| prod17 | 0   | 900 |
| prod17 | 1   | 900 |
| prod17 | 2   | 902 |
| prod17 | 3   | 905 |
| prod17 | 4   | 908 |
| prod17 | 5   | 910 |
| prod17 | 6   | 913 |
| prod17 | 7   | 915 |
| prod17 | 8   | 918 |
| prod17 | 9   | 920 |
| prod17 | 10  | 921 |
| prod17 | 11  | 923 |
| prod17 | 12  | 924 |
| prod17 | 13  | 925 |
| prod17 | 14  | 926 |
| prod17 | 15  | 926 |
| prod17 | 16  | 927 |
| prod17 | 17  | 926 |
```

Abbildung 16 - Beispiel Fehler im Report

Der zugehörige Report, s. Abbildung 16, machte sichtbar, dass im Vergleich zu den tatsächlichen Werten in der Datenbank dieser Weg keine korrekten und genauen Daten lieferte. Die Nachfragehistorie wurde pauschal als fehlerfrei ausgewiesen, ohne dass die ausgegebenen Werte zuverlässig aus der Datenbank stammten. Diese Beobachtung bildete die Rückkopplung des Design Cycle und veranlasste, die prompt-eingebettete Abfrage zu prüfen und die Datenextraktion anschließend in ein deterministisch arbeitendes Tool zu verlagern.

Beide Beispiele verdeutlichen das angewandte iterative Vorgehen: Jede Überarbeitung reagierte auf ein konkret beobachtetes Defizit und überführte das Artefakt schrittweise in einen Zustand, der korrekt wirkende Ergebnisse liefert. Anhand der folgenden Tabelle 6 ist dargestellt, wie die einzelnen Iterationsschritte zu Veränderungen geführt haben.

Agent/Aufgabe	Beschreibung ursprünglicher Prompt	Ergebnis (Problem)	Beschreibung neuer Prompt	Neues Ergebnis (Lösung)
prepare_history	Datenextraktion nur im Prompt eingebettet.	Keine korrekten Werte in der Nachfragezeitreihe.	Datenextraktion per externes deterministisches Tool.	Korrekte Werte in der Nachfragezeitreihe.
prepare_history	Keine Prüfgroßen, wie Zählung der Zeilen in Nachfragereihe.	Keine Nachvollziehbarkeit, ob vollständig oder unvollständig.	Prüfgrößen im Prompt hinzugefügt.	Nachvollziehbarkeit zum Überprüfen der Nachfragezeitreihe.
data_preparer	Verbindliche Regeln waren nicht vorhanden.	Unvollständige Nachfragezeitreihen und auch teilweise keine korrekten Werte.	Explizite, strenge verbindliche Regeln wurden hinzugefügt (s. constraints).	Vollständige und korrekte Nachfragezeitreihen.
run_diagnostics	Aufgabe war, nur die Nachfragezeitreihe zu analysieren und klassifizieren.	Unvollständige und teils auch nicht vorhandene Nachfragezeitreihen wurden klassifiziert.	Ausschließlich vollständige Nachfragezeitreihen sind zu klassifizieren.	Unvollständige und teils nicht vorhandene Nachfragezeitreihen wurden seltener klassifiziert.
diagnostics_analyst	Verbindliche Regeln waren nicht vorhanden.	Unvollständige und teils nicht vorhandene Nachfragezeitreihen wurden seltener klassifiziert.	Klassifikation nur auf Basis nachvollziehbarer Kennzahlen, diese mitliefern und entsprechende Hinweise geben.	Nur vollständige Nachfragezeitreihen wurden klassifiziert, ansonsten wurden Hinweise gegeben.
create_forecast	Nur Forecasting-Methoden wurden explizit genannt; die einzelnen Produkte nicht.	Teilweise wurden nur drei Produkte statt vier in den Forecasts erwähnt.	Explizite Nennung der vier Produkte und den Methoden, außerdem Fallback bei Fehlschlag hinzugefügt.	Alle Produkte werden in Forecasts berücksichtigt, bei Fehlschlag werden Fallback-Methoden genutzt.

forecaster	Verbindliche Regeln waren nicht vorhanden.	Trotz gleicher Nachfragezeitreihen und benutzter Forecasting-Methoden waren die Ergebnisse des Forecasts sehr unterschiedlich.	Verbindliche Regeln hinzugefügt, dass nur vorgegebenen Modelle verwendet werden und keine nicht-nachvollziehbare Berechnung der Forecasts erfolgt.	Forecast Ergebnisse nachvollziehbarer, aber immer noch sehr unterschiedlich, trotz gleicher Nachfragezeitreihen und benutzter Methoden.
evaluate_management management_advisor	Diese Aufgabe und dieser Agent wurden erst zu einem späteren Zeitpunkt Teil der Crew und wurden deshalb sorgfältig auf Basis der Erkenntnisse der anderen Aufgaben/Agenten erstellt. Diese Erkenntnisse beinhalten: Explizite Nennung der einzelnen Teile der Aufgabe, verbindliche Regeln während der Aufgabe und eine übersichtliche Darstellung zur Nachvollziehbarkeit.			
report_data	Nur die Anweisung: Erstelle einen Bericht aus den vorherigen Ergebnissen und stelle ihn übersichtlich dar.	Kurze, knappe, unübersichtliche Darstellung im Report.	Explizite Nennung, was alles dargestellt werden soll und mit dem Vermerk, dass alle Einträge der jeweiligen Teilergebnisse dargestellt werden sollen.	Ausführlichere Darstellung im Report, allerdings wird die Nachfragezeitreihe nicht komplett (vier Produkte mit 240 Perioden) ausgegeben.
data_reporter	Verbindliche Regeln waren nicht vorhanden und das Ziel war nur einen strukturierten Bericht darzustellen.	Nachfragezeitreihe wurde nicht komplett dargestellt.	Verbindliche Regeln hinzugefügt und auch im Ziel explizit hinzugefügt, dass alles vollständig dargestellt werden soll.	Nachfragezeitreihe wird trotzdem nicht komplett dargestellt.

Tabelle 6 - Darstellung Iteratives Vorgehen

4.2.6 Lessons Learned

Während der Entwicklung und Verfeinerung des Artefakts wurden einige Erfahrungen im Umgang mit KI-Agenten in CrewAI gemacht. In der nachfolgenden Tabelle 7 sind wichtige Erfahrungen sowie abgeleitete Empfehlungen dargestellt, sodass der Umgang mit den KI-Agenten verständlicher gemacht werden soll und zukünftigen Entwicklern Startwissen mitgegeben werden kann.

Typ	Erkenntnis	Auswirkung	Maßnahme / Empfehlung
Verhalten	Nicht-Determinismus: Gleicher Dateninput erzeugt bei wiederholter Ausführung nie exakt gleiche Outputs (Inhalt und Darstellung variieren).	Ergebnisse nur eingeschränkt reproduzierbar; ein Einzellauf ist nicht repräsentativ.	Stochastizität bei der Interpretation mitdenken; Auswertung über mehrere Durchläufe (50 Reports) statt Einzelfall
Verhalten	Schwankende Durchlaufzeit: Laufzeit variiert stark (ca. 2–3 bis ca. 10 Min.); einzelne Durchläufe blieben augenscheinlich hängen.	Laufzeit kein verlässlicher Indikator; Durchläufe können hängen bleiben und Beobachter weiß nicht, ob es weitergeht.	Timeout- und Wiederholungsmechanismen für produktiven Einsatz; abgebrochene Läufe verwerfen.
Verhalten	Fehlerhafte Datenübernahme zwischen Agenten: z. B. data_preparer weist 240 Runden aus, der forecaster prognostiziert jedoch Runde 158 bzw. 30 statt der nächsten zwei Perioden; Eindruck, dass nicht stets der vollständige Datenbestand verarbeitet wird.	Inhaltlich falsche Prognosen trotz korrekter Eingangsdaten; Ursache nicht abschließend geklärt.	Explizite Übergabe- und Bezugsregeln; offener Punkt für weitere Untersuchung.
Gestaltung	Präzise, regelbasierte Konfiguration: Je genauer Agenten und Tasks mit Regeln und expliziten Anweisungen konfiguriert sind, desto präziser das Ergebnis.	Geringerer Handlungsspielraum, weniger abweichende und fehlerhafte Ausgaben.	Knappe, eindeutige Tasks formulieren; Constraints zur Begrenzung des Spielraums einsetzen.
Gestaltung	Vorgegebene Output-Strukturen: Feste Ausgabestrukturen erhöhen die Konsistenz; Angaben sind reproduzierbar an derselben Stelle auffindbar.	Bessere Vergleichbarkeit über Durchläufe; einfachere Auswertung.	Verbindliches Report-/Output-Schema vorgeben.
Gestaltung	Längenbeschränkung der Reports: Zeilenumfang ist begrenzt; der Agent weist teils selbst aus, dass nur die ersten Runden des ersten Produkts ausgegeben werden, obwohl alle Perioden vorliegen.	Ein Report bildet nicht zwingend den vollständigen Datenbestand ab.	Vollständigkeit nicht allein aus dem Report schließen; Kontrollmechanismen einbauen und anhand von Beispielen testen.

Tabelle 7 - Darstellung von Erkenntnissen während der Entwicklung des Artefakts

4.3 Ergebnisse FF3: Bewertung und Evaluation

4.3.1 Strang 1: Deskriptive quantitative Inhaltsanalyse

Der erste Auswertungsstrang erfasst die Outputs der KI-Agenten-Architektur über alle 50 Reports hinweg, die auf identischem Input beruhen. Bereits ein erster Vergleich zeigt, dass die Reports trotz gleicher Datengrundlage deutlich voneinander abweichen und kein Report einem anderen gleicht. Weder in den ausgegebenen Prognosewerten noch in der jeweils gewählten Methodik gibt es exakte Übereinstimmungen. Die folgende Auswertung konzentriert sich auf drei Merkmale dieser Outputs: Ob die Diagnostik einen offensichtlichen Fehler hat, sowie die je Produkt prognostizierten Bedarfswerte und die vom forecaster eingesetzten Prognosemodelle. Diese Größen stehen nicht zufällig im Vordergrund, sondern bilden teils zugleich die Anknüpfungspunkte für den zweiten Auswertungsstrang, in dem die fehlerärmste Prognose und das jeweils geeignetste Modell eigenständig nachgerechnet und den Agenten-Outputs als Referenzmaßstab gegenübergestellt werden. Damit wird das Erkenntnisziel dieses Strangs, die Variabilität der Agenten-Outputs bei identischem Input deskriptiv abzubilden, an genau den Ergebnisgrößen operationalisiert, die anschließend einer benchmark-basierten Genauigkeitsbewertung unterzogen werden. Die erhobene Datengrundlage umfasst darüber hinaus weitere kodierte Merkmale, wie z. B. die dargestellte aufbereitete Nachfragezeitreihen, die korrekte Periodenzuordnung der Prognosen sowie die Passung zwischen Diagnostik und Modellwahl. Diese werden an dieser Stelle jedoch nicht im Fokus stehen und sind im Anhang B zu finden.

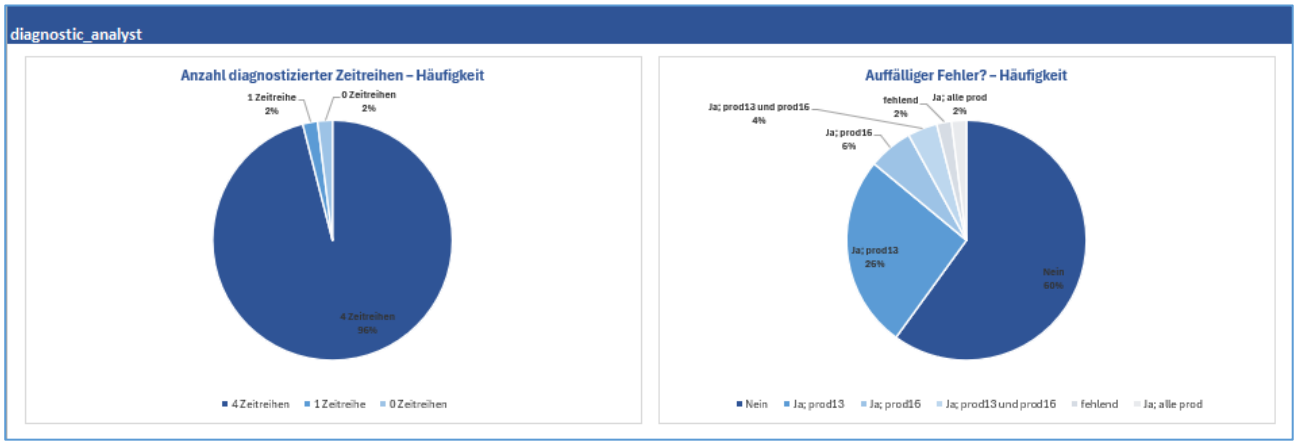


Abbildung 17 – Darstellung Häufigkeit diagnostizierter Zeitreihen und offensichtlicher Fehler

Abbildung 17 beschreibt zum einen, wie viel Nachfragezeitreihen der vier Produkte diagnostiziert wurden, wobei in 96 % der Fälle komplett diagnostiziert wurde und je in einem Fall nur eine Zeitreihe, bzw. gar keine. Das Diagramm zum auffälligen Fehler in der Diagnostik beschreibt beobachtete Fehler in Diagnsotik der jeweiligen Zeitreihen oder fehlende Zeitreihen. Für prod13 etwa attestiert der diagnostics_analyst in mehreren Läufen eine konstante Nachfrage von 420 Einheiten, obwohl das vom data_preparer als vollständig zertifizierte und an die Diagnose übergebene history_table über alle 240 Perioden deutliche Schwankungen enthält. Die Diagnose widerspricht damit ihrer eigenen, im selben Lauf bereitgestellten Datengrundlage. In rund 38 % der Reports weist die Diagnostik einen solchen auffälligen Fehler auf, etwa die Klassifikation einer nachweislich schwankenden Zeitreihe als „konstant stabil".

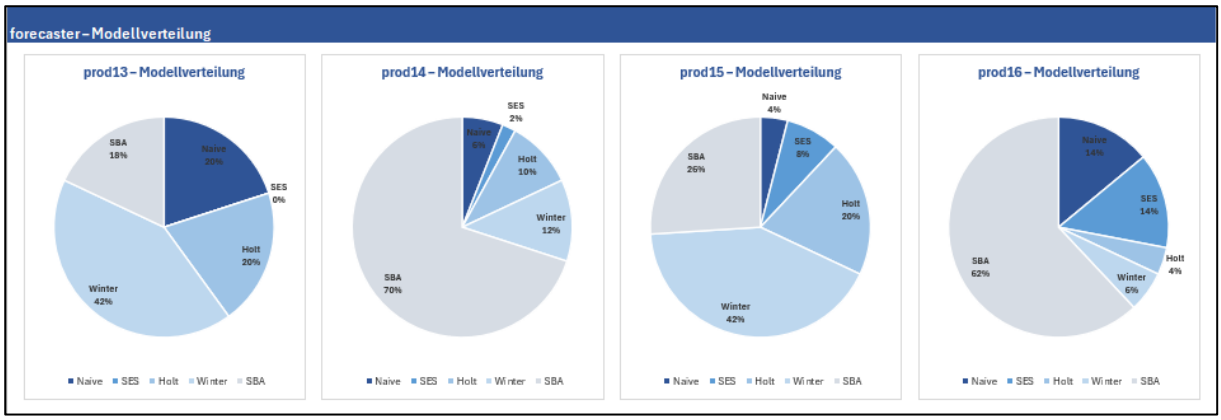


Abbildung 18 – Verteilung der benutzten Modelle zur Prognoseerstellung

Abbildung 18 zeigt die relative Häufigkeit der vom forecaster gewählten Prognosemodelle über alle 50 Reports, getrennt nach den vier Produkten. Es wird deutlich, dass für kein Produkt ein einziges Modell durchgängig gewählt wird. Vielmehr verteilt sich die Modellwahl bei identischem Input über die fünf verschiedenen Verfahren. Jedoch vergleicht man Kennzahlen der Diagnostik mit der Modellwahl für die Prognose, so passt in 94 % der Fälle die Modellwahl zur Diagnostik (s. Anhangsabbildung 8). Bei der Modellwahl bei prod13 entfällt der größte Anteil auf Winter (42 %), gefolgt von Naive und Holt (je 20 %) sowie SBA (18 %), während SES nicht auftritt (0 %). Für prod14 dominiert SBA mit 70 % deutlich, ergänzt um Winter (12 %), Holt (10 %), Naive (6 %) und SES (2 %). Bei prod15 liegt erneut Winter vorn (42 %), vor SBA (26 %), Holt (20 %), SES (8 %) und Naive (4 %). Bei prod16 wird mit 62 % überwiegend SBA gewählt, daneben Naive und SES (je 14 %), Winter (6 %) und Holt (4 %). Das jeweils am häufigsten gewählte Modell unterscheidet sich somit zwischen den Produkten: bei prod13 und prod15 ist es Winter, bei prod14 und prod16 ist es SBA.

Die Prognosewerte der 50 Reports erstrecken sich je Produkt über einen weiten Wertebereich und sind dabei nahezu durchgängig verschieden. Eine direkte Auszählung der Einzelwerte ist daher wenig aussagekräftig, stattdessen wird der Wertebereich gemäß Becker et al. (2024) klassiert (Becker, Herrmann, Heumann, Pilz, Sandor, Schäfer & Wellisch, 2024, S. 40). Die Anzahl der Klassen folgt der Regel $k = \lfloor\sqrt{n}\rfloor$ (Fahrmeir, Künstler, Pigeot und Tutz (2003), zitiert nach Becker et al., 2024, S. 41–42) und beträgt bei $n = 50$ somit $\lfloor\sqrt{50}\rfloor = 7$. Die Lage der Klassengrenzen ist durch die Quelle nicht formelhaft vorgegeben. Verlangt wird lediglich, dass die Klassen disjunkt sind und den gesamten Wertebereich überdecken (Becker et al., 2024, S. 40–41). Die Grenzen werden daher als bewusste, an der Streuung der Daten über mehrere Größenordnungen orientierte Festlegung gewählt: rund, den Wertebereich vollständig überdeckend und so, dass die Verteilungsform erkennbar bleibt. Wobei ungleich breite Klassen zulässig sind. Da sowohl die Klassenanzahl als auch die Wahl der Grenzen die Interpretation der Häufigkeitsverteilung beeinflussen können (Becker et al., 2024, S. 42), werden die verwendeten Klassen in folgender Abbildung inklusive der Häufigkeiten abgebildet:

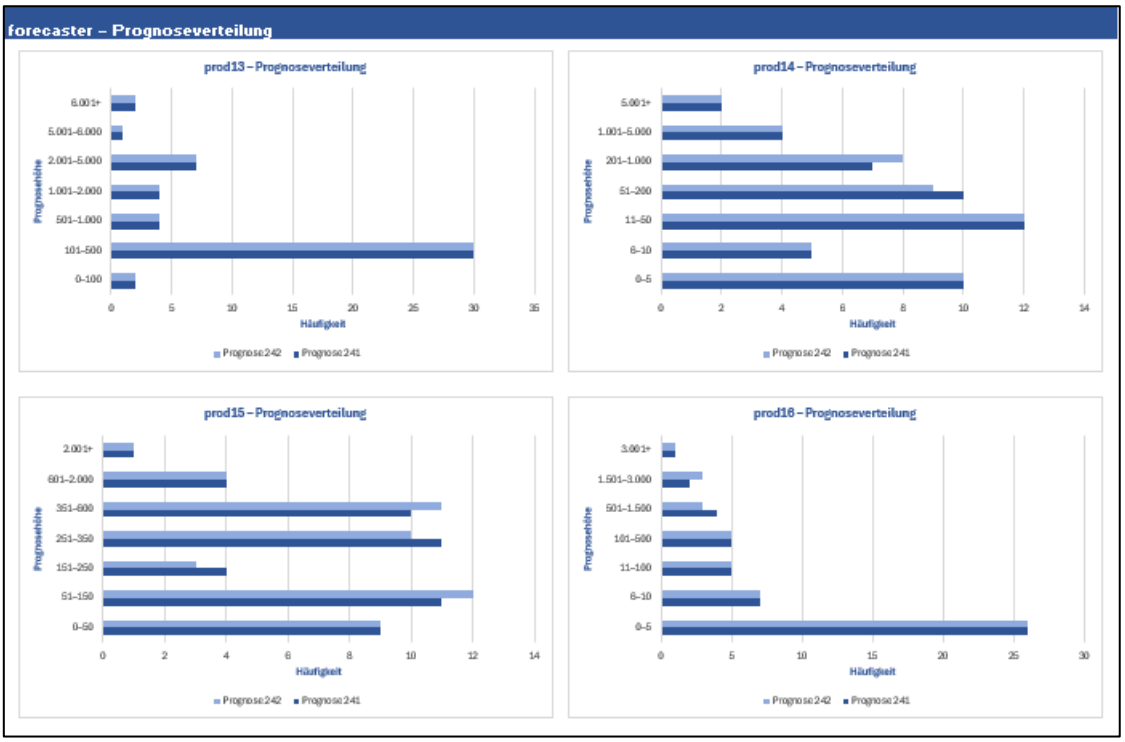


Abbildung 19 – Verteilung der Prognosewerte

Abbildung 19 stellt die Häufigkeitsverteilung der prognostizierten Bedarfswerte je Produkt über die sieben gebildeten Klassen dar, getrennt für die beiden Prognoseperioden 241 und 242. Über alle vier Produkte hinweg verlaufen die Verteilungen der beiden Perioden nahezu deckungsgleich. Bei prod13 konzentrieren sich die Prognosen deutlich in der Klasse 101–500, auf die rund 30 der Reports entfallen. Die übrigen Werte verteilen sich dünn besetzt über die höheren Klassen bis hin zu 6.001+. Die Verteilung von prod14 häuft sich in den unteren Klassen, mit den höchsten Besetzungen in 11–50 sowie in 0–5 und 51–200, während die Häufigkeiten zu den oberen Klassen hin bis 5.001+ abnehmen. Für prod15 zeigt sich eine vergleichsweise breite Verteilung mit mehreren ähnlich stark besetzten Klassen (51–150, 251–350 und 351–600) neben der Klasse 0–50, einer schwächer besetzten Klasse 151–250 und nur einer einzelnen Nennung in 2.001+. Bei prod16 fällt die Konzentration am stärksten aus: Der überwiegende Teil der Prognosen liegt in der untersten Klasse 0–5 (rund 26 Reports), während die Besetzung zu den höheren Klassen hin rasch abnimmt und in 3.001+ nur noch eine Nennung verbleibt.

4.3.2 Strang 2: Benchmark-basierte Genauigkeitsbewertung

Im zweiten Auswertungsstrang werden die Prognosen der KI-Agenten einem unabhängig berechneten Referenzmaßstab gegenübergestellt. Hierzu wurde auf Grundlage derselben Nachfragezeitreihen in einem Tabellenkalkulationsprogramm für jede der zur Verfügung stehenden Prognosemethoden die optimale Parametrisierung bestimmt: Die methodenspezifischen (Glättungs-)Parameter wurden so gewählt, dass der über die gesamte Zeitreihe gemessene Prognosefehler minimal wird. Mit diesen optimalen Parametern wurde anschließend je Methode eine Prognose berechnet. Diejenige Methode mit dem geringsten Prognosefehler bildet die fehlerärmste und damit, im Sinne dieses Strangs, optimale Prognose und dient als Referenz. Dieser Referenzmaßstab wird den Prognosen der 50 Reports gegenübergestellt. Der Vergleich erfolgt in den nachfolgenden Schritten in zwei Hinsichten: zum einen hinsichtlich der Methodenwahl, also ob die Agenten, die auf Basis der Diagnostik der Nachfragezeitreihe auch die tatsächlich fehlerärmste Methode gewählt haben, und zum anderen hinsichtlich der Abweichung der von den Agenten ausgegebenen Prognosewerte von der am häufigsten benutzten Methode zum Referenzwert der gleichen Methode.

Ergebnis-Übersicht für Prod13				
Methode	**MSE**	**RMSE**	**Prognose Runde 241**	**Prognose Runde 242**
Naive	5247630,126	2290,77064	0	0
SES	**2779297,543**	1667,122534	999,1165334	999,1165334
Holt	2973180,229	1724,291225	823,7347425	727,0763196
Holt-Winters	3064801,542	1750,65746	1391,201995	594,0888479
SBA	3462480,614	1860,774197	1004,680646	1004,680646

Abbildung 20 - Darstellung Referenzwerte für Produkt 13

Wie in Abbildung 20 zu sehen, weist für prod13 im Benchmark die Methode SES den geringsten MSE und somit den niedrigsten Prognosefehler aller berechneten Methoden auf und bildet damit die fehlerärmste Referenz (Referenzprognose 999,12 in beiden Runden). Hinsichtlich der Modellwahl zeigt sich, dass die Agenten, die ihre Methode auf Grundlage der Diagnostik der Nachfragezeitreihe wählen, das tatsächlich fehlerärmste Modell nicht getroffen haben: SES wurde von keinem der 50 Reports gewählt (Trefferquote 0 %). Am häufigsten kam stattdessen Winter (42 %) zum Einsatz, gefolgt von Naive und Holt (je 20 %) sowie SBA (18 %). Dabei ist zu berücksichtigen, dass die Referenz allein über die Minimierung des Prognosefehlers bestimmt wird, während die Agenten einer diagnostikbasierten Auswahllogik folgen: die fehlerärmste Methode muss daher nicht mit der diagnostisch nahegelegten übereinstimmen.

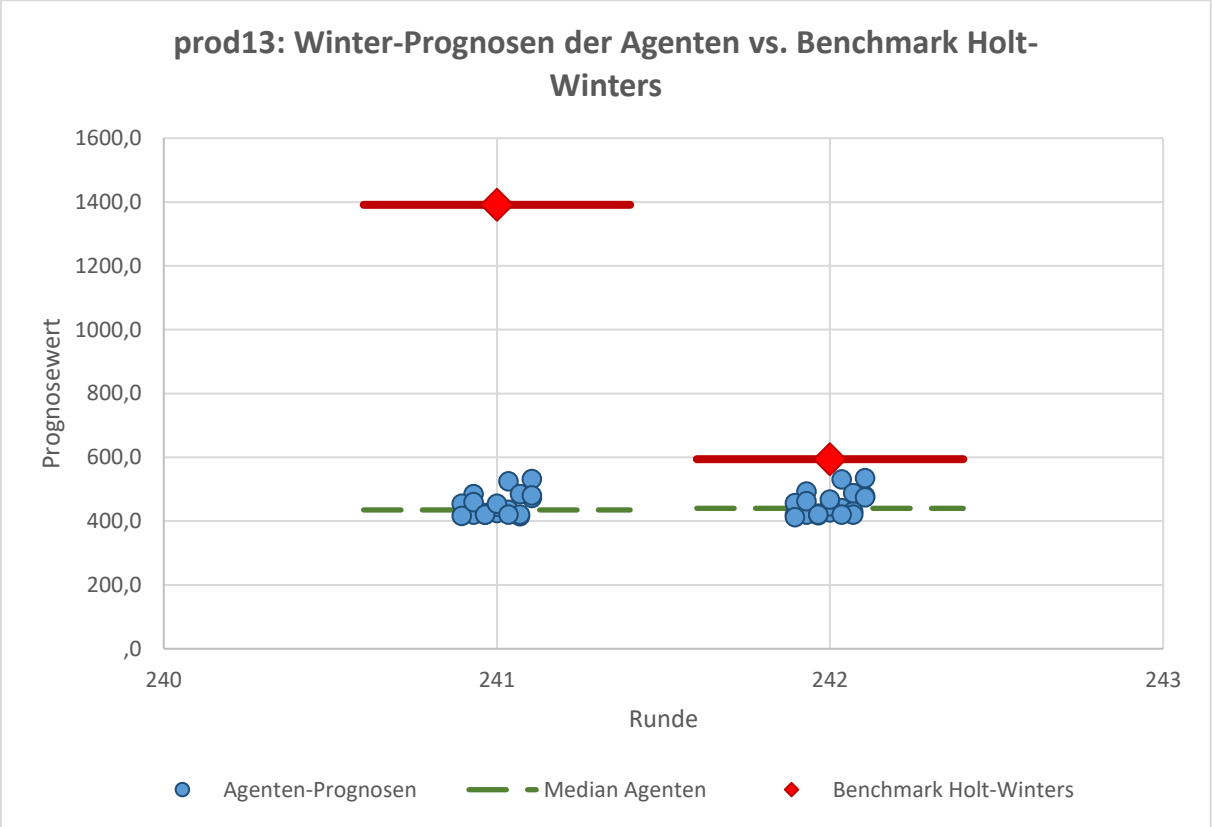


Abbildung 21 - Gegenüberstellung Prognosewerte der am häufigsten verwendeten Methode und dem Referenzwert derselben Methode für Produkt 13

Abbildung 21 betrachtet die Abweichung der Prognosewerte für das am häufigsten gewählten Modell von der Referenz desselben Modells. Hierzu wird die Methode Winter, die im Benchmark der Methode Holt-Winters entspricht, der gleichen, jedoch optimal parametrisiert nachgerechneten Modellprognose gegenübergestellt. Die nachgerechnete Holt-Winters-Prognose beträgt 1.391,20 für Runde 241 und 594,09 für Runde 242. Die unter Winter erstellten Agenten-Prognosen liegen demgegenüber eng beieinander (Median 435 in Runde 241 bzw. 440 in Runde 242, Spanne 412 bis 535)

und unterschreiten in allen 21 Fällen die Referenzwerte beider Runden deutlich. Auffällig ist zudem, dass die Agenten zwischen den beiden Runden kaum differenzieren, während die nachgerechnete Holt-Winters-Prognose von Runde 241 zu Runde 242 zurückgeht. Selbst bei identischer Methode weichen die Agenten-Prognosen somit erkennbar von dem optimalen Referenzwert ab.

Betrachtet man die Prognosen der restlichen Produkte und vergleicht diese innerhalb desselben Schemas, dann sind zwei durchgängige Muster erkennbar. Daher werden die exakten Werte und Schaubilder im Anhang nachgereicht und hier nur kurz erläutert. Erstens ist in allen vier Produkten SES die fehlerärmste Methode, wird von den diagnostikbasierten Agenten jedoch nur selten gewählt (Trefferquote zwischen 0 % und 14 %). Am häufigsten kamen stattdessen Winter (prod13, prod15) bzw. SBA (prod14, prod16) zum Einsatz. Zweitens unterschreiten die Agenten-Prognosen auch bei identischer Methode den optimal parametrisierten Referenzwert systematisch: Der Median liegt jeweils deutlich unter dem Benchmark, und nahezu alle Werte liegen darunter. Vereinzelt treten extreme Überschüsse auf (prod14: 7.200/7.500; prod15: 5.300/5.600), während prod16 die Unterprognose ohne jeden Ausreißer in besonders klarer Form zeigt. Angesichts der kleinen Produktbasis sind diese Befunde als Muster über die vier evaluierten Produkte zu verstehen und nicht zu verallgemeinern.

4.3.3 Strang 3: Qualitative Inhaltsanalyse nach Mayring

Abschließend werden die 50 Reports qualitativ bewertet. Tabelle 8 zeigt den Auszug an Dimensionen aus dem Codebuch, wobei die entsprechenden Textauszüge der Reports mittels einer Ordinalskala: hoch/mittel/niedrig/fehlend eingestuft werden.

Bereich	Nr.	Dimension	Kürzel	Anwendung
A – produktspezifisch	1	Nutzen der Prognose für das Management	MN	je Produkt (13–16)
A – produktspezifisch	2	Konkrete Handlungsempfehlung	HE	je Produkt (13–16)
A – produktspezifisch	3	Eignung des Prognoseverfahrens	EV	je Produkt (13–16)
A – produktspezifisch	4	Risiken / Unsicherheiten / Einschränkungen	RI	je Produkt (13–16)
A – produktspezifisch	5	Einschätzung zur Belastbarkeit	BE	je Produkt (13–16)
B – übergreifend	6	Kritische Methodenbewertung	KM	je Report
B – übergreifend	7	Risikoanalyse	RA	je Report
B – übergreifend	8	Gesamturteil / Empfehlungen	GE	je Report

Tabelle 8 - Zusammenfassender Auszug aus dem Codebuch

Die folgende Abbildung 22 fasst die Kodierung aller 50 Reports zusammen. Für jede der acht Dimensionen ist ausgewiesen, wie häufig die Ausprägungen hoch, mittel, niedrig und fehlend vergeben wurden. Jede Zeile bezieht sich auf n = 50 Reports und summiert sich entsprechend zu 100 %.

Produkt/Kategorie	Dimension	hoch	mittel	niedrig	fehlend	Summe
Produkt 13	MN	80,0%	16,0%	4,0%	0,0%	100,0%
Produkt 13	HE	32,0%	58,0%	10,0%	0,0%	100,0%
Produkt 13	EV	28,0%	38,0%	14,0%	20,0%	100,0%
Produkt 13	RI	12,0%	62,0%	24,0%	2,0%	100,0%
Produkt 13	BE	32,0%	38,0%	2,0%	28,0%	100,0%
Produkt 14	MN	48,0%	46,0%	6,0%	0,0%	100,0%
Produkt 14	HE	20,0%	76,0%	4,0%	0,0%	100,0%
Produkt 14	EV	28,0%	42,0%	8,0%	22,0%	100,0%
Produkt 14	RI	2,0%	70,0%	26,0%	2,0%	100,0%
Produkt 14	BE	22,0%	52,0%	2,0%	24,0%	100,0%
Produkt 15	MN	78,0%	14,0%	6,0%	2,0%	100,0%
Produkt 15	HE	20,0%	76,0%	2,0%	2,0%	100,0%
Produkt 15	EV	34,0%	36,0%	6,0%	24,0%	100,0%
Produkt 15	RI	6,0%	36,0%	52,0%	6,0%	100,0%
Produkt 15	BE	16,0%	54,0%	0,0%	30,0%	100,0%
Produkt 16	MN	52,0%	36,0%	10,0%	2,0%	100,0%
Produkt 16	HE	16,0%	78,0%	4,0%	2,0%	100,0%
Produkt 16	EV	26,0%	42,0%	12,0%	20,0%	100,0%
Produkt 16	RI	4,0%	44,0%	48,0%	4,0%	100,0%
Produkt 16	BE	16,0%	54,0%	0,0%	30,0%	100,0%
Übergreifend	KM	18,0%	70,0%	12,0%	0,0%	100,0%
Übergreifend	RA	26,0%	68,0%	6,0%	0,0%	100,0%
Übergreifend	GE	34,0%	44,0%	22,0%	0,0%	100,0%
Summe		28,3%	50,0%	12,2%	9,6%	100,0%

Abbildung 22 - Abbildung relativer Häufigkeiten der qualitativen Inhaltsanalyse der Management Reports

Über alle 1.150 Einzelkodierungen hinweg dominiert die mittlere Qualitätsstufe mit 50,0 %, gefolgt von hoch (28,3 %), niedrig (12,2 %) und fehlend (9,6 %). Die Bewertungen des management_advisor sind damit weit überwiegend vorhanden und inhaltlich nachvollziehbar, erreichen aber seltener die höchste Stufe konkreter, produktspezifischer Aussagekraft. Das Gesamtbild ist eine deutliche Häufung im mittleren Band. Auffällig ist die Verteilung der fehlenden Abschnitte: Von den 110 als fehlend kodierten Fällen entfallen rund 90 % auf lediglich zwei Dimensionen: die Eignung des Prognoseverfahrens (EV) und die Einschätzung zur Belastbarkeit (BE). BE fehlt je nach Produkt in 24 % bis 30 % der Reports, EV in 20 % bis 24 %. Demgegenüber sind der Managementnutzen (MN) und die Handlungsempfehlung (HE) praktisch immer vorhanden (fehlend ≤ 2 %), und die drei übergreifenden Dimensionen (KM, RA, GE) fehlen in keinem einzigen Report. Der Agent liefert die produktübergreifende Synthese also zuverlässig, lässt jedoch auf Produktebene am ehesten die Belastbarkeits- und Verfahrenseinschätzung aus.

Auf Ebene der einzelnen Dimensionen zeigen sich folgende Besonderheiten. Beim Managementnutzen (MN) ist der Anteil der Stufe hoch durchgehend ausgeprägt, variiert aber zwischen den Produkten: prod13 (80 %) und prod15 (78 %) liegen deutlich über prod14 (48 %) und prod16 (52 %). Die Handlungsempfehlung (HE) ist dagegen über alle Produkte hinweg klar von der mittleren Stufe geprägt (58 % bis 78 %). Die Verfahrenseignung (EV) verteilt sich breiter und erreicht in 26 % bis 34 % der Fälle die Stufe hoch, aber fällt in rund einem Fünftel der Reports ganz aus. Bei den Risiken (RI) wird die Stufe hoch nur selten erreicht (2 % bis 12 %), zugleich steigt der Anteil generischer Risikoaussagen (niedrig) für prod15 (52 %) und prod16 (48 %) stark an, während er für prod13 (24 %) und prod14 (26 %) niedriger liegt. Die Belastbarkeitseinschätzung (BE) ist nahezu zweigeteilt: Sie wird entweder auf mittlerer Stufe vergeben oder fehlt ganz, während die Stufe niedrig praktisch nicht vorkommt (0 % bis 2 %). Bei den übergreifenden Dimensionen überwiegt ebenfalls die mittlere Stufe

(KM 70 %, RA 68 %, GE 44 %), das Gesamturteil (GE) ist dabei am stärksten polarisiert, mit zugleich dem höchsten hoch-Anteil (34 %) und dem höchsten niedrig-Anteil (22 %) dieser drei Dimensionen.

Die hier dargestellten Häufigkeiten sind zunächst rein deskriptiv zu verstehen. Eine inhaltliche Einordnung der Muster, wie etwa der Zusammenhang zwischen Nachfragecharakteristik und Bewertungsqualität oder die systematische Auslassung der Belastbarkeitseinschätzung sowie die methodische Einschränkung der eingeschränkten Zirkularität erfolgen in der Diskussion bzw. den Limitationen.

5. Diskussion

5.1 Einordnung der Ergebnisse

Das auffälligste Ergebnis des ersten Strangs ist, dass bei identischer Datengrundlage kein Report einem anderen gleicht und sich die Modellwahl über alle fünf Forecasting-Methoden verteilt. Diese Variabilität ist jedoch nicht als Fehler der Architektur zu deuten, sondern als Ausdruck der inhärenten Stochastizität generativer Sprachmodelle, die als probabilistische Maschinen ihre Ausgaben nicht deterministisch erzeugen (Jannelli et al., 2026, S. 15). Davon zu unterscheiden ist die Halluzination als Erzeugung inhaltlich falscher Informationen (Taulli & Deshmukh, 2025, S. 256). Beide Phänomene treten in der Architektur auf: Neben der reinen Streuung, die lediglich die Schwankung der Ausgaben bei identischem Input bezeichnet, zeigt sich in rund 38 % der Reports eine inhaltlich falsche Diagnose, die der mitgelieferten Datengrundlage widerspricht und damit eine Halluzination im engeren Sinne darstellt. Auffallend ist, dass diese Falschaussage entsteht, obwohl dem diagnostics_analyst das offenbar vollständige history_table über das context-Feld bereitsteht: Die Datengrundlage ist korrekt verfügbar, wird aber nicht faktentreu genutzt. Über die regeltreue Verarbeitungskette (94 % Passung) pflanzt sich der Fehler bis in formulierte Managementaussagen fort. Die Streuung begründet die in FF1 abgeleitete Anforderung der Output-Stochastizität (Nr. 13), die Diagnose-Halluzination verweist zusätzlich auf die Grenzen der reinen context-Bindung ohne deterministische Faktenprüfung.

Der zweite Strang ist hinsichtlich zweier voneinander zu trennender Befunde zu deuten. Erstens treffen die diagnostikbasiert gewählten Modelle das im Benchmark fehlerärmste Verfahren nur selten (Trefferquote 0 % bis 14 %). Diese Abweichung ist jedoch kein Indikator für eine fehlerhafte Modellwahl. Der Referenzmaßstab wird ausschließlich über die Minimierung des Prognosefehlers über die gesamte Zeitreihe bestimmt, während die Agenten einer diagnostikgeleiteten Auswahllogik anhand der Zeitreihencharakteristik folgen. Beide Logiken müssen nicht zum selben Modell führen, sodass die Differenz methodisch erwartbar ist und nicht als Versagen interpretiert werden darf. Diese Entlastung gilt jedoch nur, soweit die zugrunde liegende Diagnose zutrifft. Da die Diagnose in rund 38 % der Fälle falsch ist, beruht ein Teil der Modellwahlen auf einer falschen Annahme, sodass die Abweichung im Vergleich zum Benchmark hier nicht mehr allein mit der unterschiedlichen Auswahllogik zu erklären ist. Aussagekräftiger ist der zweite Befund: Selbst bei identischer Methode unterschreiten die Prognosen den optimal parametrisierten Referenzwert systematisch und differenzieren kaum zwischen den beiden Prognoseperioden, während die nachgerechnete Referenz zwischen den Runden variiert. Hier zeigt sich eine Genauigkeitsschwäche, die sich unmittelbar an eine Grenze generativer KI zurückbinden lässt: Large Language Models arbeiten bei streng formalen Aufgaben wie mathematischen Berechnungen unzuverlässig, da neuronale Netze grundsätzliche Schwierigkeiten mit exakter Logik aufweisen (Ertel, 2016, S. 74). Werden Prognosewerte sprachlich generiert statt deterministisch berechnet, sind plausibel wirkende, aber von der korrekten Berechnung abweichende Werte die zu erwartende Folge. Insofern lässt sich der Befund auch als numerische Entsprechung des Halluzinationsphänomens verstehen (Taulli & Deshmukh, 2025, S. 256). Damit bestätigt die Evaluation empirisch die in FF1 theoriegeleitet abgeleitete Anforderung des Tool-Grounding (Nr. 6), also die Trennung von sprachlichem Schlussfolgern und deterministischer Berechnung.

Der dritte Strang fügt sich in dasselbe Muster. Die Bewertungen des management_advisor sind weit überwiegend vorhanden und nachvollziehbar (mittlere Stufe 50,0 %, hoch 28,3 %), erreichen aber seltener die höchste Stufe konkreter, produktspezifischer Aussagekraft. Aufschlussreich ist die Verteilung der Auslassungen: Rund 90 % der als fehlend kodierten Fälle entfallen auf die Eignung des Prognoseverfahrens und die Belastbarkeitseinschätzung, während die produktübergreifende Synthese aus kritischer Methodenbewertung, Risikoanalyse und Gesamturteil in keinem Report fehlt. Der Agent liefert also zuverlässig die narrativ-synthetische Leistung, lässt aber am ehesten die technisch-spezifische Einzelbewertung aus. Dies entspricht dem Befund aus Strang 2: Zuverlässig ist die Architektur bei der narrativ-synthetischen Aufbereitung und der regelkonformen Strukturierung. Unzuverlässig ist sie nicht nur bei formal-numerischen Aufgaben, sondern auch bei der faktentreuen Begründung ihrer analytischen Aussagen an den Daten, wie der Diagnosefehler zeigt. Auffällig ist zudem eine Produktabhängigkeit. Der Managementnutzen erreicht bei prod13 (80 %) und prod15 (78 %) deutlich häufiger die höchste Stufe als bei prod14 (48 %) und prod16 (52 %), also bei jenen Produkten, deren Nachfrage überwiegend mit der für sporadische Verläufe geeigneten Syntetos-Boylan-Approximation prognostiziert wird. Dieser Zusammenhang ist allerdings doppeldeutig. Gerade prod13, welches die höchste Management-Qualität erreicht, ist zugleich das Produkt mit den häufigsten Diagnosefehlern. Die konfident formulierte, aber falsche Konstanz-Diagnose erzeugt offenbar gerade jene konkreten, planungsbezogenen Aussagen, die das Kategoriensystem als hoch einstuft. Die hohe Bewertungsqualität misst damit die rhetorisch-strukturelle Form der Aussage, nicht ihre Faktentreue. Eine inhaltlich falsche, aber konkret formulierte Aussage kann die höchste Stufe erreichen. Beide Deutungen, der Zusammenhang mit der Nachfragecharakteristik wie auch der Effekt der teils falschen Diagnose, sind angesichts der kleinen Produktbasis als Auffälligkeit und nicht als gesicherter Befund zu werten.

Zusammenfassend ergibt sich ein konsistentes Bild zur Beantwortung von FF3. Die KI-Agenten-Architektur unterstützt das Forecasting vor allem dort, wo sprachlich-kognitive Fähigkeiten gefragt sind: bei der Aufbereitung und Vollständigkeitsprüfung der Datenbasis, der regelkonformen, nachvollziehbaren Methodenwahl sowie der Aufbereitung der Ergebnisse für das Management. In diesen Schritten liegt der erkennbare Mehrwert. Die eigentliche numerische Prognose hingegen ist ohne deterministische Absicherung weder reproduzierbar (erster Strang) noch hinreichend genau (zweiter Strang). Der Nutzen der Agenten liegt damit gegenwärtig in der Unterstützung des Forecasting-Prozesses und nicht in der autonomen, präzisen Prognose. Dieser Befund deckt sich mit der in FF1 abgeleiteten Anforderung eines Human-in-the-Loop (Nr. 16).

5.2 Einordnung gegenüber Literaturstand

Die zwei der drei Schwächen der Architektur, die nicht deterministische Output-Erzeugung und die Genauigkeitsschwäche bei der numerischen Prognose, bestätigen die Gestaltungslogik der Arbeiten, die generative KI im Forecasting bereits erfolgreich eingesetzt haben. Abbaszadeh Nakhost et al. (2025) begegnen den ersten beiden Problemen konstruktiv: Sie lassen das Modell ausführbaren Code erzeugen, der lokal deterministisch ausgeführt wird (Abbaszadeh Nakhost et al., 2025, S. 6–7), und reduzieren die Output-Stochastizität durch Mittelung über wiederholte Läufe (Abbaszadeh Nakhost et al., 2025, S. 11–12). Die vorliegende Evaluation liefert die empirische Gegenprobe, denn eine Architektur, die auf diese Absicherungen verzichtet, weist exakt die Streuung und die systematische Abweichung auf, die jene Maßnahmen verhindern sollen. In dieselbe Richtung weisen Jannelli et al. (2026), worin das LLM als probabilistische Maschine für komplexe Berechnungen als ungeeignet befunden wird und nur ein externes Tool für das Forecasting verwendet wird (Jannelli et al., 2026, S. 19). Der hier gezeigte Befund, dass sprachlich generierte Prognosewerte vom korrekt berechneten Referenzwert abweichen, fügt sich in diese Einordnung ein. Die dritte Schwäche, die teils nicht faktentreue Diagnose, adressieren diese Maßnahmen hingegen nicht unmittelbar. Sie betrifft

nicht die Berechnung, sondern die inhaltliche Begründung einer sprachlichen Aussage an den Daten und verlangt daher eine zusätzliche, deterministische Faktenprüfung.

Auch hinsichtlich der Rolle des Menschen bestätigt die Arbeit den Forschungsstand. Die aus den Befunden abgeleitete Schlussfolgerung, dass der Nutzen der Agenten in der Prozessunterstützung und nicht in der autonomen Prognose liegt, deckt sich mit dem breiten Konsens, KI-Agenten bei wirkungsstarken SCM-Entscheidungen keine volle Entscheidungsautonomie zuzugestehen (Jannelli et al., 2026, S. 18, S. 21; Trifone et al., 2026, S. 23–24). Darüber hinaus greift diese Arbeit eine konkrete Forschungsforderung auf: Song et al. (2026) bemängeln, dass viele Studien auf idealisierten oder simulierten Datensätzen beruhen und ein Vergleich mit etablierten statistischen Verfahren selten bleibt, sodass der Mehrwert von LLMs ungeklärt ist, und fordern, über Proof-of-Concept-Implementierungen hinauszugehen und robuste, vergleichende Evaluationsrahmen zu entwickeln (Song et al., 2026, S. 13). Der benchmark-basierte zweite Auswertungsstrang setzt diese Forderung um, indem er die Agenten-Prognosen einem unabhängig berechneten, fehlerminimalen Referenzmaßstab gegenüberstellt. Das Ergebnis, dass die Agenten diesen Maßstab nicht erreichen, ist dabei selbst ein Beitrag zur von Song et al. (2026) aufgeworfenen Frage nach dem tatsächlichen Mehrwert.

Die Befunde dieser Arbeit relativieren die Reichweite einzelner positiver Ergebnisse des Forschungsstands. Zouaghi et al. (2025) zeigen, dass generative KI im Forecasting bei niedriger Temperatureinstellung die besten Fehlerwerte erzielt und das Prognosemodell ARIMA übertrifft (Zouaghi et al., 2025, S. 21). Dieses Ergebnis steht nicht im Widerspruch zu den hier gezeigten Schwächen, ist aber an Bedingungen geknüpft, die die vorliegende Architektur nicht erfüllt: Während Zouaghi et al. (2025) ein gezielt konfiguriertes Einzelmodell optimieren, generiert die hier untersuchte Multi-Agenten-Architektur die Prognosewerte ohne deterministische Absicherung. Der Nutzen generativer KI im Forecasting erweist sich damit als bedingungsabhängig und nicht als unmittelbare Eigenschaft des Modells.

5.3 Beitrag für Theorie

Der theoretische Beitrag dieser Arbeit lässt sich in drei Punkten zusammenfassen. Den ersten Beitrag bildet die in FF1 erarbeitete Liste technologischer Anforderungen an KI-Agenten im Forecasting von SCM, der theoriegeleitete, literaturbeständige und literaturneue Anforderungen zu einer strukturierten Liste zusammenführt. Die Evaluation des Artefakts verstärkt einzelne dieser Anforderungen, ohne sie durchgängig zu bestätigen. Deutlich gestützt wird die Anforderung des Human-in-the-Loop (Nr. 16): Da sich die numerische Prognose als weder reproduzierbar noch hinreichend genau erweist, untermauert dies die Notwendigkeit menschlicher Kontrolle bei wirkungsstarken Prognoseentscheidungen. Verstärkt wird dies zudem durch die teils faktenwidrigen Diagnosen: Eine konfident, aber falsch formulierte Aussage, die ungeprüft in eine Entscheidungsgrundlage gelangt, untermauert den Bedarf menschlicher Kontrolle zusätzlich. Für das Tool-Grounding (Nr. 6) liefert die Evaluation lediglich einen indirekten Beleg, denn der Befund, dass rein sprachlich erzeugte Prognosewerte nicht prognosegenau sind, stützt die Forderung nach einer Trennung von sprachlichem Schlussfolgern und deterministischer Berechnung, ohne sie im Sinne eines kontrollierten Vergleichs nachzuweisen. Einen zweiten, unabhängigen Beleg liefert die teilweise Diagnose-Halluzination: Ein deterministischer Abgleich der Diagnose mit einfachen Reihenstatistiken hätte die falsche Klassifikation erkennen können. Das weitet die Forderung nach deterministischer Absicherung über die numerische Berechnung hinaus auf den Diagnoseschritt aus. Den zweiten Beitrag bildet die Architektur selbst. Im Sinne der gestaltungsorientierten Forschung liegt der wissenschaftliche Beitrag primär in der Konstruktion eines zweckmäßigen Artefakts (Hevner et al., 2004, S. 83). Das hier entwickelte Artefakt ist eine LLM-basierte Multi-Agenten-Architektur, die auf Basis einer Datenbank eigenständig Nachfrageprognosen erstellt, diese bewertet und als strukturierten Report ausgibt. Damit schließt die Arbeit die identifizierte Forschungslücke, da keine der vorliegenden Arbeiten die Bedarfsprognose zum zentralen Untersuchungsgegenstand eines LLM-basierten Multi-Agenten-

Systems gemacht hatte. Die Architektur erweitert die Wissensbasis somit um eine zuvor lediglich konzeptionell geforderte Instanziierung. Einen dritten, methodisch nachgeordneten Beitrag bildet das Evaluationsdesign. Da der Nicht-Determinismus von LLM-Agenten einen Einzellauf unrepräsentativ macht, greift eine an einzelnen Outputs ausgerichtete Bewertung zu kurz. Die Kombination aus deskriptiver Variabilitätsanalyse (Strang 1), benchmark-basierter Genauigkeitsbewertung (Strang 2) und skalierender strukturierender Inhaltsanalyse nach Mayring (Strang 3) stellt einen übertragbaren Ansatz bereit, stochastische Agenten-Outputs zugleich entlang formaler, genauigkeitsbezogener und qualitativer Dimensionen zu beurteilen.

5.4 Beitrag für Praxis

Aus den Befunden lassen sich konkrete Gestaltungs- und Einsatzempfehlungen für die betriebliche Praxis ableiten. Die zentrale Empfehlung betrifft die Trennung von sprachlicher und numerischer Verarbeitung. Da die rein sprachlich erzeugten Prognosewerte weder reproduzierbar noch hinreichend genau sind, sollte die eigentliche Berechnung nicht dem Sprachmodell überlassen, sondern an ein deterministisch arbeitendes Tool delegiert werden. Daraus folgt eine realistische Erwartungshaltung: Die Architektur ist als Ergänzung des Forecasting-Prozesses zu verstehen und nicht als Ersatz der eigentlichen Prognose. Ihren erkennbaren Mehrwert entfaltet sie dort, wo hauptsächlich sprachlich-kognitive Fähigkeiten gefragt sind, also bei der Aufbereitung und Vollständigkeitsprüfung der Daten, der regelkonformen, nachvollziehbaren Methodenwahl und insbesondere der adressatengerechten, produktübergreifenden Aufbereitung der Ergebnisse für das Management, die der Agent zuverlässig liefert. Produktspezifische und technische Einzelaussagen, etwa zur Eignung des Verfahrens oder zur Belastbarkeit der Prognose, werden hingegen am ehesten ausgelassen oder wirken generisch und sind daher vor einer Verwendung zu prüfen. Auch vorhandene, selbstbewusst formulierte Diagnosen und Bewertungen sind vor einer Verwendung gegen die tatsächlichen Daten zu prüfen, da sie der Datengrundlage widersprechen können. Zwei Vorkehrungen ergänzen diese Empfehlungen. Erstens bleibt ein Human-in-the-Loop erforderlich, denn wirkungsstarke Prognoseentscheidungen bedürfen einer menschlichen Genehmigung und Eingriffsmöglichkeit, solange die Outputs nicht vollständig verlässlich und erklärbar sind. Zweitens ist die Stochastizität der Agenten-Outputs einzuplanen, da ein Einzellauf nicht repräsentativ ist. Für den Einsatz empfiehlt sich daher eine Auswertung über mehrere Läufe sowie eine feste, reproduzierbare Output-Struktur. Über diese anwendungsbezogenen Empfehlungen hinaus bietet die im Rahmen der iterativen Artefaktentwicklung erstellte Lessons-Learned-Tabelle (Tabelle 7) Entwicklern ein praxisnahes Startwissen für den Aufbau vergleichbarer Multi-Agenten-Systeme.

6. Limitationen und zukünftige Forschung

6.1 Methodische/inhaltliche Grenzen

Auf methodischer Ebene ist die Anlage der qualitativen Inhaltsanalyse einzuschränken. Das Kategoriensystem wurde teils induktiv aus dem Material gebildet und nach einem ersten Materialdurchgang überarbeitet, jedoch ohne die von Mayring (2014) vorgesehene Reliabilitätsprüfung. Als optimales Vorgehen wäre demnach mindestens eine Intracoder-Prüfung angezeigt gewesen, bei der das Material nach einem zeitlichen Abstand erneut kodiert und mit der ersten Kodierung verglichen wird und geringe Abweichungen als Stabilitätsmaß ausgewiesen werden (Mayring, 2014, S. 111). Da die Bewertung in Strang 3 auf einer Ordinalskala erfolgt, wäre die Übereinstimmung über einen für ordinale Daten geeigneten Koeffizienten wie Krippendorffs Alpha zu quantifizieren gewesen (Mayring, 2014, S. 112–114). Diese Prüfschritte wurden nicht durchgeführt, sodass die qualitative Auswertung in diesem Punkt hinter dem nach Mayring (2014) optimalen Vorgehen zurückbleibt. Eine zweite Einschränkung der qualitativen Analyse betrifft ihre eingeschränkte Zirkularität. Mayring zufolge müssen die Strukturierungsdimensionen einer deduktiven Kategorienanwendung aus der Fragestel-

lung abgeleitet und theoretisch begründet sein (Mayring, 2014, S. 95–97). Im vorliegenden Fall leiten sich die acht Strukturierungsdimensionen jedoch aus der Soll-Struktur des management_report ab, also aus der Aufgabenkonfiguration des bewerteten Agenten selbst, und nicht aus einem davon unabhängigen theoretischen Hintergrund. Dadurch entsteht eine teilweise Zirkularität. Denn das Kategoriensystem, das die Reports bewertet, ist zugleich die Operationalisierung dessen, was der Agent erzeugen soll, sodass das Vorhandensein der Dimensionen teilweise misst, ob der Agent seiner eigenen Vorgabe folgt. Damit fehlt ein vom Artefakt unabhängiges externes Kriterium, an dem die Bewertung im Sinne der Konstrukt- bzw. korrelativen Validität abgesichert werden könnte (Mayring, 2014, S. 111). Begrenzt wird diese Zirkularität dadurch, dass nicht die Dimensionen, sondern die Ausprägungen (hoch, mittel, niedrig, fehlend) induktiv aus dem Material der 50 Reports gewonnen wurden und Qualitätsabstufungen über das bloße Vorhandensein hinaus erfassen. Die Zirkularität beschränkt sich somit auf die Herkunft der Dimensionen und nicht auf die Qualitätsbewertung innerhalb der Dimensionen.

Eine weitere Grenze betrifft den Stichprobenumfang von 50 Reports. Ursprünglich war vorgesehen, das Verhalten der Architektur über 100 Durchläufe bei jeweils identischem Input zu untersuchen. Bereits nach 50 Durchläufen zeigte sich jedoch, dass kein Report einem anderen gleicht, sodass diese Ausgangsüberlegung verworfen wurde und damit zusätzlich der Vergleich mit einer Referenzprognose sowie eine qualitative Inhaltsanalyse in den Vordergrund gerückt sind. Die Zahl der Reports wurde auf 50 als hinreichend große Stichprobe reduziert. Zudem wurde mit CrewAI ein Framework pauschal gewählt, ohne alternative Agenten-Frameworks systematisch zu vergleichen. CrewAI war dabei das zunächst naheliegende Werkzeug. Auch die Überlegung, das eingesetzte Sprachmodell gegen ein anderes Modell zu vergleichen, wurde verworfen, da das herangezogene Vergleichsmodell o3 von OpenAI bei gleicher Konfiguration in acht von zwölf Versuchen abstürzte beziehungsweise Reports ohne eine einzige Information lieferte. Hinzu treten zwei weitere methodische Einschränkungen. Zum einen wurde kein kontrollierter Vergleich mit und ohne deterministisches Tool-Grounding durchgeführt. Da der forecaster die Prognosewerte ohne ein solches Tool erzeugte, fällt das inhärente Limit des Sprachmodells mit der konkreten Umsetzungsentscheidung zusammen, sodass die Schlussfolgerung zur Bedeutung des Tool-Grounding gestützt, aber nicht unabhängig nachgewiesen ist. Zum anderen ist die empirische Literaturbasis der ersten Forschungsfrage schmal: Die systematische Recherche lieferte im Schnittfeld nur wenige relevante Treffer, sodass mehrere Anforderungen des Katalogs auf jeweils nur einer Quelle beruhen und der Katalog quellenseitig begrenzt belegt ist.

Inhaltlich ist festzuhalten, dass die in FF1 erarbeiteten Anforderungen im Artefakt nur teilweise und nicht vollständig umgesetzt wurden, wie etwa das Tool-Grounding der numerischen Berechnung. Ferner wurde mit GPT-4.1-mini ein Sprachmodell pauschal gewählt, ohne die Auswahl anhand von Kennzahlen oder vergleichbaren Kriterien zu begründen. Zudem erfolgt die Methodenwahl der Agenten diagnostikbasiert anhand der Zeitreihencharakteristik, während in der Praxis moderne Systeme die geeignete Methode über die Minimierung des Prognosefehlers selektieren. Eine weitere Grenze betrifft den Modellpool. Die Beschränkung auf fünf klassische Zeitreihenverfahren ist zwar eine bewusste Designentscheidung, da diese Verfahren die diagnostizierten Zeitreihencharakteristika abdecken und zugleich als nachvollziehbare Referenzmethoden der Evaluation dienen. Weitere Verfahren, wie etwa kausale, maschinell lernende oder generativ-KI-native Ansätze, bleiben unberücksichtigt. Hinzu kommen zwei Einschränkungen aus der Evaluation. Erstens erfasst das Kategoriensystem des dritten Strangs die formale und rhetorische Qualität der Bewertungen, nicht ihre Faktentreue. Denn auf falschen Diagnosen beruhende, aber konkret formulierte Aussagen können die höchste Ausprägung erreichen, s. prod13 mit größtem Managementnutzen, aber auch größter Fehlerauffälligkeit in der Diagnose. Zweitens verfügt die Architektur über keinen Faktenkonsistenz-

Check auf der Diagnose-Ebene: Anders als der data_preparer, dessen Vollständigkeitsregel Datenfehler an der Quelle abfängt, kann die Diagnose ihrer eigenen Datengrundlage widersprechen und sich ungeprüft durch die Verarbeitungskette fortpflanzen.

6.2 Zukünftige Forschung

Aus den dargestellten Grenzen und Befunden ergeben sich zwei wesentliche Richtungen für die zukünftige Forschung. Die erste betrifft die Weiterentwicklung des Artefakts selbst. Ansatzpunkt ist die Optimierung der Architektur mit dem Ziel, die Genauigkeit der Prognosen zu erhöhen. Naheliegend hierzu ist die Verwendung von Tools zur Berechnung der Prognosen. Ebenso ließe sich der Diagnoseschritt durch einen deterministischen Plausibilitätscheck absichern, der die Diagnose gegen Kennzahlen prüft und das Tool-Grounding damit eine Stufe früher, bereits in der Mustererkennung, einbaut. Darüber hinaus sind weitergehende empirische Untersuchungen sinnvoll, etwa ein systematischer Vergleich verschiedener Sprachmodelle, der im Rahmen dieser Arbeit aufgrund technischer Probleme des Vergleichsmodells nicht durchgeführt werden konnte. Schließlich bietet die Optimierung der Prompts auf Ebene der Agenten und Aufgaben weiteres Verbesserungspotenzial, da sich bereits in der iterativen Verfeinerung gezeigt hat, dass eine präzisere, regelbasierte Konfiguration die Qualität und Konsistenz der Outputs erhöht.

Die zweite Richtung betrifft die Ausweitung des Multi-Agenten-Systems. Erweist sich der hier untersuchte Forecasting-Ansatz als tragfähig, lässt sich die Crew über die Bedarfsprognose hinaus auf weitere Bereiche des Supply Chain Managements übertragen, etwa auf Produktion, Beschaffung oder Distribution, sodass ein durchgängiges, arbeitsteiliges Agentensystem entlang der Lieferkette entsteht. Im Kontext des BPOS-Games wäre zudem zu untersuchen, wie sich das System verhält, wenn es nicht nur Entscheidungsgrundlagen liefert, sondern mit autonomen Handlungsentscheidungen wie Bestell- und Produktionsentscheidungen verbunden wird. Eine solche Erweiterung würde den im vorliegenden Prototyp bewusst gewahrten Human-in-the-Loop schrittweise zurücknehmen und erlauben, die Ergebnisse einer zunehmend autonomen Steuerung empirisch zu beobachten und zu bewerten.

7. Fazit

Die Bedarfsprognose bildet die informatorische Grundlage der Planung im SCM, stößt mit steigender Komplexität, fehleranfälliger Datenbasis und kaum beherrschbarer Informationsmenge jedoch zunehmend an die Grenzen klassischer, manuell gestützter Verfahren. Generative KI in Form LLM-basierter Multi-Agenten-Systeme verspricht hier Entlastung, war bislang aber nicht mit der Bedarfsprognose als zentralem Untersuchungsgegenstand erforscht. Vor diesem Hintergrund verfolgte die vorliegende Arbeit das Ziel, ein prototypisches Artefakt zu entwickeln und zu evaluieren: eine Architektur LLM-basierter KI-Agenten für das Forecasting im SCM auf Basis historischer Nachfragedaten. Methodisch folgte sie der gestaltungsorientierten Forschung und gliederte sich entlang dreier Forschungsfragen, deren Beantwortung im Folgenden zusammengeführt wird. Die erste Forschungsfrage nach den technologischen Anforderungen an KI-Agenten im Supply Chain Management wurde über eine systematische Literaturrecherche beantwortet. Durch die Verknüpfung eines Sieben-Schichten-Modells generativer KI-Agenten mit den Schritten des Forecasting-Prozesses entstand eine Liste von sechzehn Anforderungen, die theoriegeleitete, literaturbeständige und literaturneu abgeleitete Anforderungen zu einer strukturierten Übersicht zusammenführt. Auf dieser Grundlage beantwortete die zweite Forschungsfrage die Frage nach einer geeigneten Architektur. Als Antwort entstand eine in CrewAI realisierte Fünf-Agenten-Architektur, die an die Datenbank eines betrieblichen Planspiels angebunden ist und deren Designentscheidungen jeweils an den Anforderungen des Katalogs begründet sind. Dieses instanziierte Artefakt bildet den primären wissenschaftlichen Beitrag der Arbeit. Die dritte Forschungsfrage, inwieweit KI-Agenten das Forecasting unter-

stützen können, wurde anhand von 50 bei identischem Input erzeugten Reports entlang dreier aufeinander bezogener Stränge evaluiert. Die Auswertung ergibt ein konsistentes Bild: Die Outputs streuen erheblich, kein Report gleicht einem anderen, und die sprachlich generierten Prognosewerte unterschreiten den unabhängig nachgerechneten, optimal parametrisierten Referenzwert systematisch. Dazu kommt, dass ein Teil der Diagnosen der eigenen Datengrundlage widerspricht. Die managementorientierte, produktübergreifende Aufbereitung gelingt demgegenüber zuverlässig in ihrer Form, ohne dass damit die inhaltliche Korrektheit der zugrunde liegenden Diagnosen gesichert ist, während technisch-spezifische Einzelbewertungen am ehesten ausgelassen werden. Der Nutzen der Architektur liegt damit gegenwärtig in der Unterstützung des Forecasting-Prozesses und in der Entlastung der anfangs genannten Last: der Aufbereitung und Strukturierung einer kaum noch übersehbaren Informationsmenge. Nicht jedoch in der autonomen, präzisen Prognose. Der Mehrwert der Arbeit ist damit ein doppelter. Theoretisch schließt sie die identifizierte Forschungslücke, indem sie erstmals die Bedarfsprognose zum zentralen Gegenstand eines LLM-basierten Multi-Agenten-Systems macht und die zuvor lediglich konzeptionell geforderte Instanziierung bereitstellt. Ergänzt wird dies um die Anforderungsliste sowie um ein übertragbares, dreistränginges Evaluationsdesign, das stochastische Agenten-Outputs zugleich entlang formaler, genauigkeitsbezogener und qualitativer Dimensionen beurteilt. Praktisch leitet die Arbeit konkrete Gestaltungsempfehlungen ab: die Trennung sprachlichen Schlussfolgerns von deterministischer Berechnung durch Tool-Grounding, eine realistische Erwartungshaltung gegenüber der Architektur als Ergänzung statt Ersatz der eigentlichen Prognose sowie die Notwendigkeit eines Human-in-the-Loop bei wirkungsstarken Prognoseentscheidungen. Künftige Arbeiten können an zwei Stellen ansetzen. Zum einen lässt sich das Artefakt selbst weiterentwickeln, insbesondere durch die deterministische Berechnung der Prognosen über externe Tools, durch den systematischen Vergleich verschiedener Sprachmodelle und durch eine weiter verfeinerte, regelbasierte Konfiguration der Agenten und Aufgaben. Zum anderen lässt sich das arbeitsteilige Agentensystem über die Bedarfsprognose hinaus auf weitere Bereiche der Lieferkette übertragen und schrittweise mit autonomeren Entscheidungskompetenzen ausstatten. Bis dahin empfiehlt sich die Architektur für die betriebliche Praxis als unterstützendes Assistenzsystem, dessen Ergebnisse den menschlichen Entscheider entlasten, ohne ihn zu ersetzen.

8. Literaturverzeichnis

Abbaszadeh Nakhost, M., Bloemer, A. & Minner, S. (2025). Full automation or just a copilot? A study of large language models on spare parts demand forecasting with OpenAI o1-mini. *International Journal of Production Research*, 10.1080/00207543.2025.2591225

Arndt, H. (2021). *Supply Chain Management: Optimierung logistischer Prozesse* (8. Aufl.). Wiesbaden, Deutschland: Springer Gabler.

Becker, T., Herrmann, R., Heumann, C., Pilz, S., Sandor, V., Schäfer, D. & Wellisch, U. (2024). *Stochastische Risikomodellierung und statistische Methoden: Angewandte Stochastik für die aktuarielle Praxis* (2. Aufl.). Berlin, Heidelberg: Springer Spektrum. https://doi.org/10.1007/978-3-662-69532-6

Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., . . . Raffel, C. (2021). *Extracting training data from large language models*. Vortrag auf der 30th USENIX security symposium (USENIX Security 21), Abgerufen am 14. Juni 2026 von https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting

Chopra, S. & Meindl, P. (2019). *Supply Chain Management: Strategy, Planning, and Operation* (7. Aufl.). Boston, MA: Pearson Education.

Christopher, M. (2023). *Logistics and supply chain management* (6. Aufl.). Harlow, England: Pearson Education.

Dincelli, E. & Jafarian, H. (2025). Exploring the agentic metaverse's potential for transforming cybersecurity workforce development. *MIS Quarterly Executive*, 24(4), 317–330.·https://doi.org/10.17705/2msqe.00122

Döring, N. (2016). *Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften* (5. Aufl.). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-41089-5

Education 4 Success. (o. J.). *The BPOS game*. Abgerufen am 17.06.2026 von https://e4success.net/bpos-business-process-optimization-simulation-game/

Ertel, W. (2016). *Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung* (4. Aufl.). Wiesbaden, Germany: Springer Vieweg. https://doi.org/10.1007/978-3-658-13549-2

Ferreira, A. C. A., Francisco, M. B. & De Pinho, A. F. (2025). The use of artificial intelligence in supply chain management: Systematic literature review and future research directions. *IEEE Access*, 13, 157828–157841.·https://doi.org/10.1109/ACCESS.2025.3603866

Feuerriegel, S., Hartmann, J., Janiesch, C. & Zschech, P. (2024). Generative AI. *Business & information systems engineering*, 66(1), 111–126.·https://doi.org/10.1007/s12599-023-00834-7

Hendriksen, C. (2023). Artificial intelligence for supply chain management: Disruptive innovation or innovative disruption? *Journal of Supply Chain Management*, 59(3), 65–76.·https://doi.org/10.1111/jscm.12304

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2), 87–92.·https://doi.org/.

Hevner, A. R., March, S. T., Park, J. & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1), 75–105.·https://doi.org/10.2307/25148625

Huang, K. (Hrsg.). (2025). *Agentic AI: Theories and Practices*. Cham, Schweiz: Springer.

Hyndman, R. J. & Athanasopoulos, G. (2021). *Forecasting: principles and practice* (3. Aufl.). Melbourne, Australien: OTexts. Abgerufen am 12. Juni 2026 von https://otexts.com/fpp3/

Jackson, I., Ivanov, D., Dolgui, A. & Namdar, J. (2024). Generative artificial intelligence in supply chain and operations management: A capability-based framework for analysis and implementation. *International Journal of Production Research*, 62(17), 6120–6145.·https://doi.org/10.1080/00207543.2024.2309309

Jannelli, V., Schoepf, S., Bickel, M., Netland, T. H. & Brintrup, A. (2026). Agentic LLMs in the supply chain: Towards autonomous multi-agent consensus-seeking. *International Journal of Production Research*, 10.1080/00207543.2025.2604311

Klaus, P. & Müller, S. (Hrsg.). (2012). *The Roots of Logistics: A Reader of Classical Contributions to the History and Conceptual Foundations of the Science of Logistics*. Heidelberg, Germany: Springer.

Li, L., Liu, Y., Jin, Y., Cheng, T. E. & Zhang, Q. (2024). Generative AI-enabled supply chain management: The critical role of coordination and dynamism. *International Journal of Production Economics*, 277, Article 109388.·https://doi.org/10.1016/j.ijpe.2024.109388

Li, L., Zhu, W., Chen, L. & Liu, Y. (2024). Generative AI usage and sustainable supply chain performance: A practice-based view. *Transportation Research Part E: Logistics and Transportation Review*, 192, Article 103761.·https://doi.org/10.1016/j.tre.2024.103761

Mayring, P. (2014). *Qualitative content analysis: Theoretical foundation, basic procedures and software solution*. Klagenfurt.

McCarthy, J., Minsky, M. L., Rochester, N. & Shannon, C. E. (2006). A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence. *AI magazine*, 27(4), 12–14.·https://doi.org/10.1609/aimag.v27i4.1904

Organisation for Economic Co-operation and Development. (2024). *Explanatory memorandum on the updated OECD definition of an AI system* (Bericht Nr. 8). Abgerufen am 14. Juni 2026 von https://www.oecd.org/content/dam/oecd/en/publications/reports/2024/03/explanatory-memorandum-on-the-updated-oecd-definition-of-an-ai-system_3c815e51/623da898-en.pdf

OWASP. (2024). *OWASP Top 10 for LLM applications 2025* (Abgerufen am 14. Juni 2026 von https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/

Panja, S., Zheng, T. & Glock, C. H. (2026). Potentials of generative AI in logistics and supply chain management: A guiding review on applications and future research directions. *International Journal of Production Research*, 10.1080/00207543.2026.2680586

Petropoulos, F., Apiletti, D., Assimakopoulos, V., Babai, M. Z., Barrow, D. K., Taieb, S. B., . . . Boylan, J. E. (2022). Forecasting: theory and practice. *International Journal of forecasting*, 38(3), 705–871.·https://doi.org/10.1016/j.ijforecast.2021.11.001

Quan, Y., Liu, Z., Benaben, F. & Montreuil, B. (2026). Leveraging large language models to enhance multi-agent risk assessment in supply chain networks. *International Journal of Production Research*, 10.1080/00207543.2026.2619562

Richey, R. G., Jr., Chowdhury, S., Davis-Sramek, B., Giannakis, M. & Dwivedi, Y. K. (2023). Artificial intelligence in logistics and supply chain management: A primer and roadmap for research. *Journal of Business Logistics*, 44(4), 532–549.·https://doi.org/10.1111/jbl.12364

Shukla, N. & Kiridena, S. (2016). A fuzzy rough sets-based multi-agent analytics framework for dynamic supply chain configuration. *International Journal of Production Research*, 54(23), 6984–6996.·https://doi.org/10.1080/00207543.2016.1151567

Song, Z., Xie, Y., Yang, L. & Zhao, Y. (2026). Large language models in supply chain management: A systematic literature review and application framework. *International Journal of Production Research*, 10.1080/00207543.2026.2641103

Syntetos, A. A. & Boylan, J. E. (2005). The accuracy of intermittent demand estimates. *International Journal of forecasting*, 21(2), 303–314.·https://doi.org/10.1016/j.ijforecast.2004.10.001

Taulli, T. & Deshmukh, G. (2025). *Building Generative AI Agents: Using LangGraph, AutoGen, and CrewAI*. New York, NY: Apress. https://doi.org/10.1007/979-8-8688-1134-0

Trifone, N., Dallari, F. & Brintrup, A. (2026). Drivers and constraints of agentic AI in supply chain management: an exploratory study. *International Journal of Production Research*, 10.1080/00207543.2026.2669878

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236), 433–460.·https://doi.org/10.1093/mind/LIX.236.433

Webster, J. & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS quarterly*, 26(2), xiii–xxiii.·https://doi.org/10.2307/4132319

Xu, L., Mak, S. & Brintrup, A. (2021). Will bots take over the supply chain? Revisiting agent-based supply chain automation. *International Journal of Production Economics*, 241, Article 108279.·https://doi.org/10.1016/j.ijpe.2021.108279

Xu, L., Mak, S., Minaricova, M. & Brintrup, A. (2024). On implementing autonomous supply chains: A multi-agent system approach. *Computers in Industry*, 161, Article 104120.·https://doi.org/10.1016/j.compind.2024.104120

Zouaghi, I., Beldjoudi, S., Gunasekaran, A., Fosso Wamba, S. & Sahrane, S. (2025). Balancing performance and innovation in AI-driven supply chains through temperature-scaled hallucination control. *International Journal of Production Research*, 10.1080/00207543.2025.2601264

Übersicht verwendeter Hilfsmittel

Produktname	**Bezugsquelle**	**Genutzter Funktionsumfang im Rahmen dieser Arbeit**
Claude (Anthropic)	Anthropic, claude.ai	Rechercheunterstützung, Prüfung und Verfeinerung eigener Textentwürfe, Kalibrierung von Quellen- und Zitationsvorschlägen, Unterstützung bei der Strukturierung der Arbeit sowie bei der Aufbereitung von Excel-Dokumenten
ChatGPT (OpenAI)	OpenAI, chatgpt.com	Rechercheunterstützung
GitHub Copilot (in Visual Studio Code)	GitHub (Microsoft), github.com/features/copilot	Erläuterungen im Umgang mit CrewAI und seiner Funktionsweise sowie Vorschläge und schrittweise Hinweise zur Implementierung der Agenten-Crews
Anara	Anara (anara.com)	KI-gestützte Extraktion und Aufbereitung von Quelltexten / PDF-Inhalten
DeepL	DeepL SE, deepl.com	Übersetzung englischsprachiger Texte ins Deutsche
EndNote 25 (Clarivate)	Clarivate; Campuslizenz der HNU	Literaturverwaltung: automatisierte Erstellung der Quellenbelege und des Literaturverzeichnisses

Anhang

Anhang A - Quelltext des PostgreSQLReaderTool

Das PostgreSQLReaderTool ist das deterministisch arbeitende Werkzeug, über das der data_preparer die Nachfragehistorie aus der operativen Datenbank des BPOS-Games extrahiert. Es kapselt die in custom_tool.py hinterlegte demand_history_query, die orders und order_amounts verknüpft, fehlende Werte als nullwert markiert, Mehrfacheinträge aggregiert und über einen CROSS JOIN für jede Produkt-Runden-Kombination genau einen Datensatz erzeugt. Der vollständige Quelltext belegt die Reproduzierbarkeit der Datenextraktion.

```python
class PostgreSQLReaderToolInput(BaseModel):
    """Input schema for PostgreSQL Reader Tool."""
    host: Optional[str] = Field(None, description="Der PostgreSQL Host (z.B. 'localhost' oder IP-Adresse). Falls nicht angegeben, wird DB_HOST aus .env verwendet.")
    port: Optional[int] = Field(None, description="Der Port der PostgreSQL-Datenbank (Standard: 5432). Falls nicht angegeben, wird DB_PORT aus .env verwendet.")
    database: Optional[str] = Field(None, description="Der Name der Datenbank. Falls nicht angegeben, wird DB_NAME aus .env verwendet.")
    user: Optional[str] = Field(None, description="Der Datenbank-Benutzer. Falls nicht angegeben, wird DB_USER aus .env verwendet.")
    password: Optional[str] = Field(None, description="Das Passwort für den Benutzer. Falls nicht angegeben, wird DB_PASSWORD aus .env verwendet.")
    table: Optional[str] = Field(None, description="Der Name der Tabelle, aus der gelesen werden soll. Falls nicht angegeben, wird DB_TABLE aus .env verwendet.")
    query: Optional[str] = Field(None, description="Optionale SQL-Query. Wenn nicht angegeben, wird SELECT * FROM table verwendet.")

class PostgreSQLReaderTool(BaseTool):
    name: str = "PostgreSQL Reader Tool"
    description: str = (
        "Liest Daten aus einer PostgreSQL-Datenbank. "
        "Kann entweder eine komplette Tabelle oder das Ergebnis einer benutzerdefinierten SQL-Query abrufen. "
        "Für demand_history erzeugt das Tool deterministisch ein vollständiges history_table "
        "für die Produkte prod13, prod14, prod15 und prod16 über alle aktuell in orders "
        "vorhandenen Runden mit round >= 0. Für jede Produkt-Runden-Kombination wird genau "
        "ein Datensatz geliefert. Fehlende Werte werden als demand_amount = 0 und status = nullwert "
        "ausgegeben. Mehrfache Einträge derselben Produkt-Runden-Kombination werden aggregiert. "
        "Gibt die Daten als JSON-String zurück, geeignet für Datenverarbeitung. "
        "WICHTIG: Stelle sicher, dass die Datenbank erreichbar ist und die Credentials korrekt sind."
    )
    args_schema: Type[BaseModel] = PostgreSQLReaderToolInput
```

Anhangsabbildung 1 - Quelltext PostgreSQLReaderTool 1/6

```python
def _run(
    self,
    host: Optional[str] = None,
    port: Optional[int] = None,
    database: Optional[str] = None,
    user: Optional[str] = None,
    password: Optional[str] = None,
    table: Optional[str] = None,
    query: Optional[str] = None
) -> str:
    conn = None
    try:
        # Fallback zu Environment-Variablen wenn Parameter nicht übergeben
        host = host or os.getenv('DB_HOST', 'localhost')
        port = port or int(os.getenv('DB_PORT', '5432'))
        database = database or os.getenv('DB_NAME')
        user = user or os.getenv('DB_USER')
        password = password or os.getenv('DB_PASSWORD')
        table = "demand_history"

        # Validierung: Kritische Parameter müssen vorhanden sein
        if not database or not user or not password:
            return "Fehler: DB_NAME, DB_USER und DB_PASSWORD müssen entweder als Parameter übergeben oder in .env gesetzt sein."

        # Verbindung zur PostgreSQL-Datenbank aufbauen
        conn = psycopg2.connect(
            host=host,
            port=port,
            database=database,
            user=user,
            password=password
        )
```

Anhangsabbildung 2 - Quelltext PostgreSQLReaderTool 2/6

```
demand_history_query = """
    WITH products(product_id) AS (
        VALUES ('prod13'), ('prod14'), ('prod15'), ('prod16')
    ),
    rounds AS (
        SELECT DISTINCT o.round::int AS round
        FROM orders o
        WHERE o.round >= 0
    ),
    raw_demand AS (
        SELECT
            oa.product_id,
            o.round::int AS round,
            oa.amount::numeric AS demand_amount
        FROM orders o
        INNER JOIN order_amounts oa ON o.id = oa.order_id
        WHERE oa.product_id IN ('prod13', 'prod14', 'prod15', 'prod16')
          AND o.round >= 0
          AND oa.supplier_id = 'tier2'
    ),
    demand_agg AS (
        SELECT
            product_id,
            round,
            SUM(demand_amount) AS demand_amount,
            COUNT(*) AS source_rows
        FROM raw_demand
        GROUP BY product_id, round
    )
```

Anhangsabbildung 3 - Quelltext PostgreSQLReaderTool 3/6

```
    SELECT
        p.product_id,
        r.round,
        COALESCE(d.demand_amount, 0) AS demand_amount,
        CASE
            WHEN d.product_id IS NULL THEN 'nullwert'
            WHEN d.demand_amount < 0 THEN 'invalid'
            WHEN d.source_rows > 1 THEN 'aggregated'
            ELSE 'ok'
        END AS status
    FROM products p
    CROSS JOIN rounds r
    LEFT JOIN demand_agg d ON d.product_id = p.product_id AND d.round = r.round
    ORDER BY p.product_id, r.round
""" # Diese Query erzeugt ein vollständiges history_table für die Produkte prod13-prod16 über alle Runden in orders mit round >= 0, markiert fehlende Werte al
    # Die finale Selektion mit CROSS JOIN stellt sicher, dass alle Kombinationen aus Produkten und Runden vorhanden sind, auch wenn es keine Bestellungen gibt
# Die demand_history_query wird in diesem Tool immer ausgeführt.
query = demand_history_query

# Daten mit pandas lesen
df = pd.read_sql(query, conn)

# Konvertiere zu Dictionary (für JSON)
data = df.to_dict(orient='records')

if table == "demand_history": # Validierung für demand_history: Überprüfe, ob alle Kombinationen aus prod13-prod16 und Runden in orders mit round >= 0 vorhand
    round_count = int(df["round"].nunique()) if not df.empty else 0
    product_count = 4
    expected_rows = product_count * round_count
    actual_rows = len(df)
    nullwert_count = int((df["status"] == "nullwert").sum()) if not df.empty else 0
```

Anhangsabbildung 4 - Quelltext PostgreSQLReaderTool 4/6

```
    if actual_rows != expected_rows:
        return (
            "Fehler: Unvollständiges history_table erzeugt. "
            f"Erwartet: {expected_rows} Zeilen, erhalten: {actual_rows}."
        )

# Rückgabe als String
if table == "demand_history": #Rückgabe mit zusätzlichen Validierungsinformationen für demand_history, damit die Agenten verstehen, dass sie ein volls
    result = {
        "data": data,
        "columns": ["product_id", "round", "demand_amount", "status"],
        "actual_rows": actual_rows,
        "rows": actual_rows,
        "round_count": round_count,
        "product_count": product_count,
        "expected_rows": expected_rows,
        "nullwert_count": nullwert_count,
        "source": "demand_history_complete",
        "host": host,
    }
else:
    result = {
        "data": data,
        "columns": list(df.columns),
        "actual_rows": len(df),
        "rows": len(df),
        "source": f"{database}.{table}",
        "host": host
    }
return json.dumps(result, ensure_ascii=False)
```

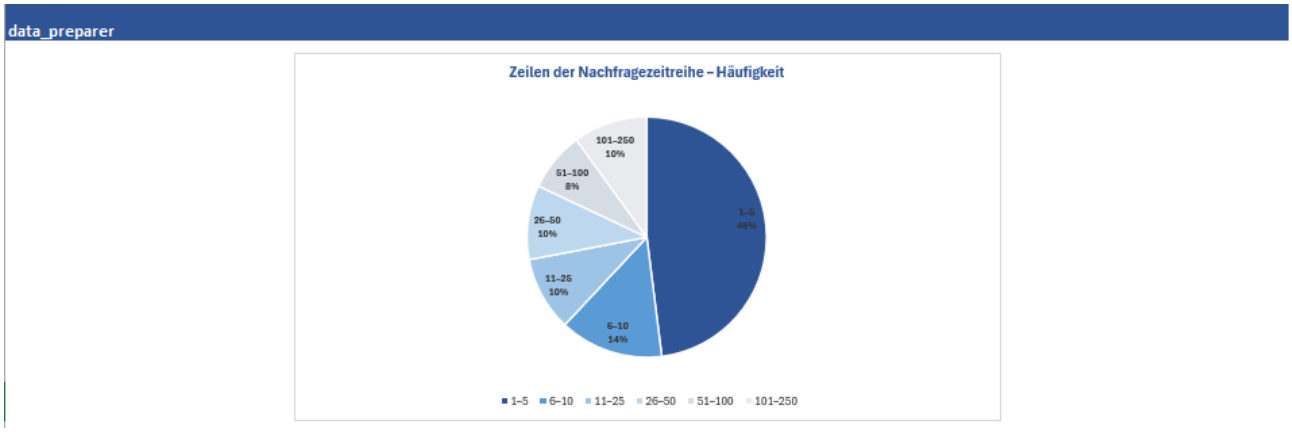
Anhangsabbildung 5 - Quelltext PostgreSQLReaderTool 5/6

```
    except psycopg2.OperationalError as e:
        return f"Fehler beim Verbinden zur PostgreSQL-Datenbank: {str(e)}. Prüfe Host, Port, Credentials."
    except psycopg2.Error as e:
        return f"Fehler bei PostgreSQL-Query: {str(e)}"
    except Exception as e:
        return f"Fehler beim Lesen der PostgreSQL-Datenbank: {str(e)}"
    finally:
        # Verbindung schließen
        if conn is not None:
            conn.close()
```

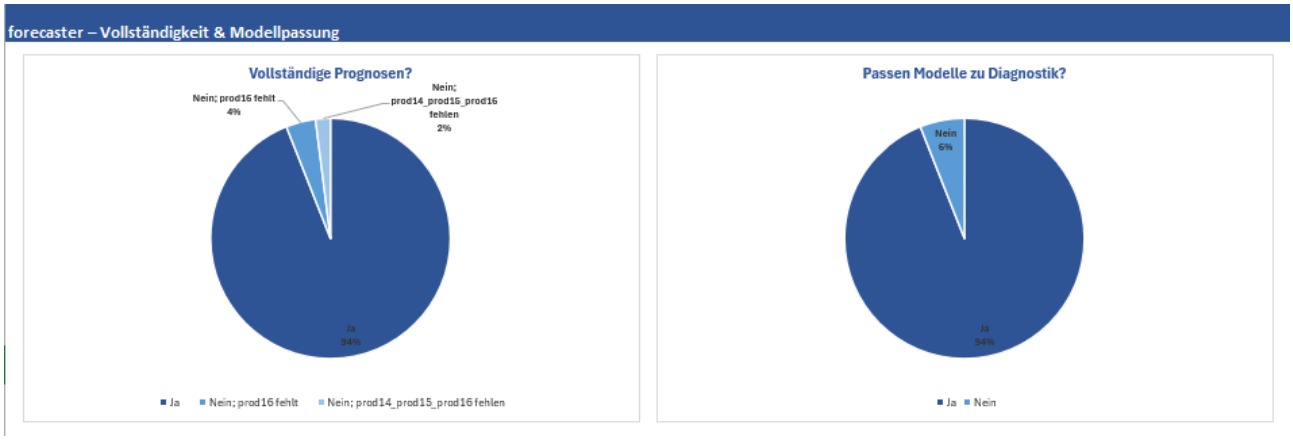
Anhangsabbildung 6 - Quelltext PostgreSQLReaderTool 6/6

Anhang B – Variabilität der Agenten Outputs

Anhang B ergänzt die deskriptive Variabilitätsanalyse des ersten Auswertungsstrangs um die kodierten Merkmale, die dort nicht im Vordergrund standen. Anhangsabbildung 7 zeigt, wie häufig die aufbereitete Nachfragezeitreihe über die 50 Reports hinweg überhaupt im Report dargestellt wird. Anhangsabbildung 8 stellt die vollständigen je Produkt prognostizierten Bedarfswerte einschließlich ihrer Periodenzuordnung sowie die Passung zwischen der Diagnostik und der gewählten Prognosemethode dar. Aus dieser Auswertung stammt die im Hauptteil berichtete Passungsquote von 94 %, die allein die Regeltreue der Verarbeitungskette und nicht die Korrektheit der Modellwahl gegenüber den tatsächlichen Daten belegt.



Anhangsabbildung 7 – Häufigkeit Darstellung Nachfragezeitreihen im Report



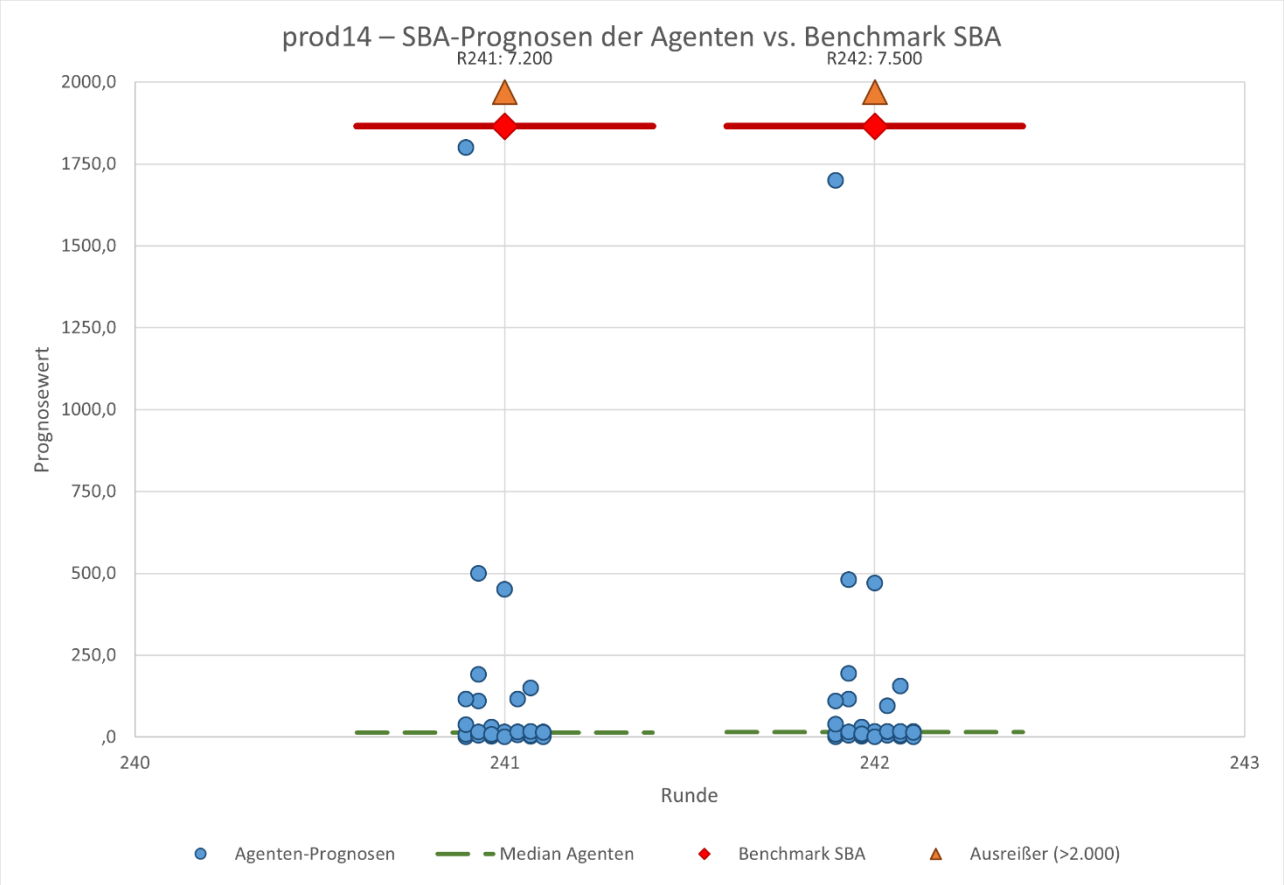
Anhangsabbildung 8 – Vollständige Prognosen und Passung zur Diagnostik

Anhang C – Referenzwerte und Gegenüberstellung der Prognosewerte der Produkte 14 bis 16

Anhang C reicht die im Hauptteil nur zusammengefasst dargestellten Detailergebnisse des zweiten Auswertungsstrangs für die Produkte 14 bis 16 nach. Für jedes Produkt weisen die Anhangsabbildungen zunächst die eigenständig nachgerechneten, optimal parametrisierten Referenzwerte aus (Anhangsabbildung 9, 11 und 13) und stellen diesen anschließend die Agenten-Prognosen desselben Modells gegenüber (Anhangsabbildung 10, 12 und 14).

Ergebnis-Übersicht für Prod14				
Methode	**MSE**	**RMSE**	**Prognose Runde 241**	**Prognose Runde 242**
Naive	7648662,795	2765,621593	0	0
SES	**4362574,12**	2088,677601	1923,01307	1923,01307
Holt	4629186,621	2151,554466	1625,725692	1425,661941
Holt-Winters	4706016,862	2169,335581	2302,247417	1397,009011
SBA	6313219,756	2512,612138	1865,156945	1865,156945

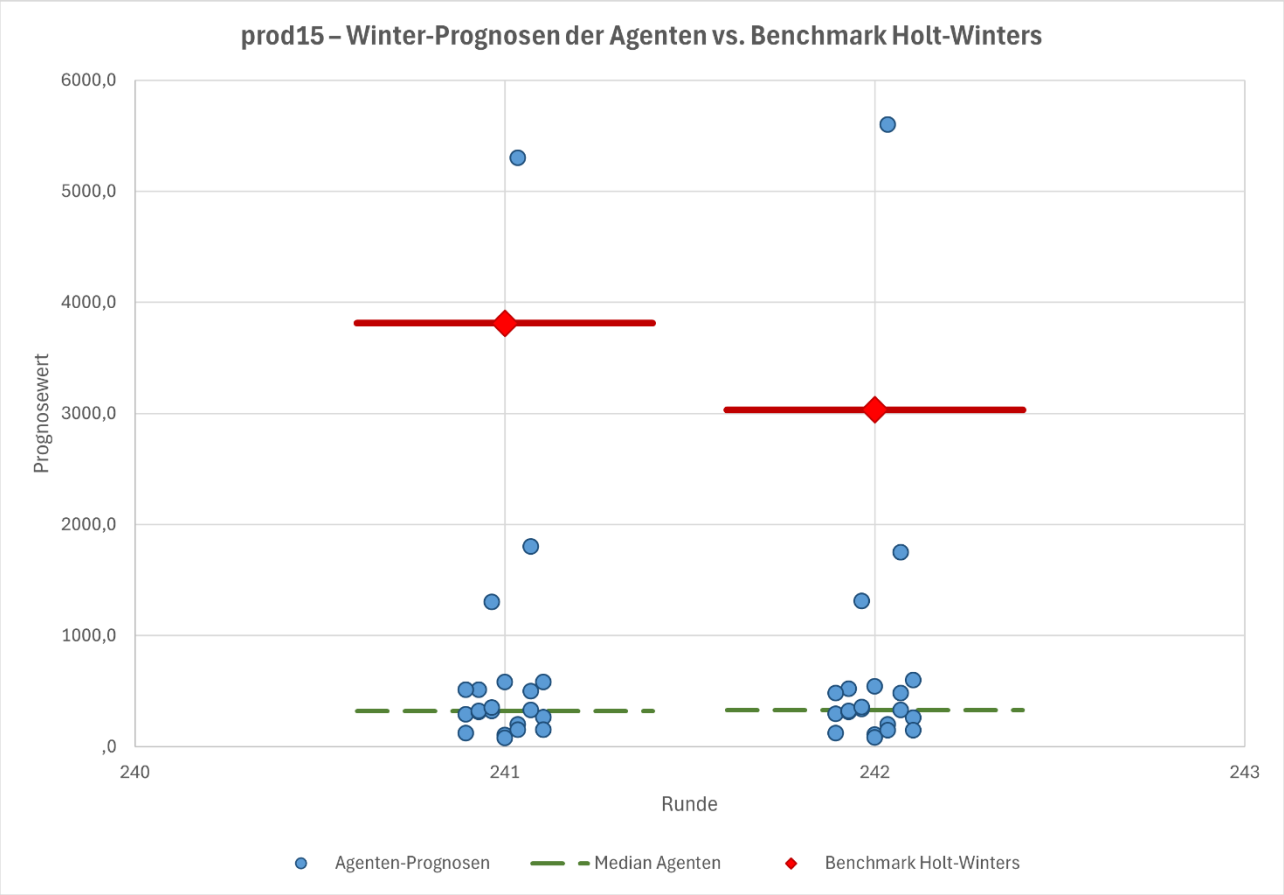
Anhangsabbildung 9 - Referenzwerte für Produkt 14



Anhangsabbildung 10 - Gegenüberstellung Referenzwert und Prognosewerte des gleichen Models (Produkt 14)

Ergebnis-Übersicht für Prod15				
Methode	**MSE**	**RMSE**	**Prognose Runde 241**	**Prognose Runde 242**
Naive	5831698,745	2414,891042	0	0
SES	**3211677,633**	1792,115407	3167,373129	3167,373129
Holt	3361248,689	1833,370854	3197,409938	3119,441283
Holt-Winters	3417274,517	1848,587168	3811,708708	3032,362287
SBA	4888744,185	2211,050471	1975,620878	1975,620878

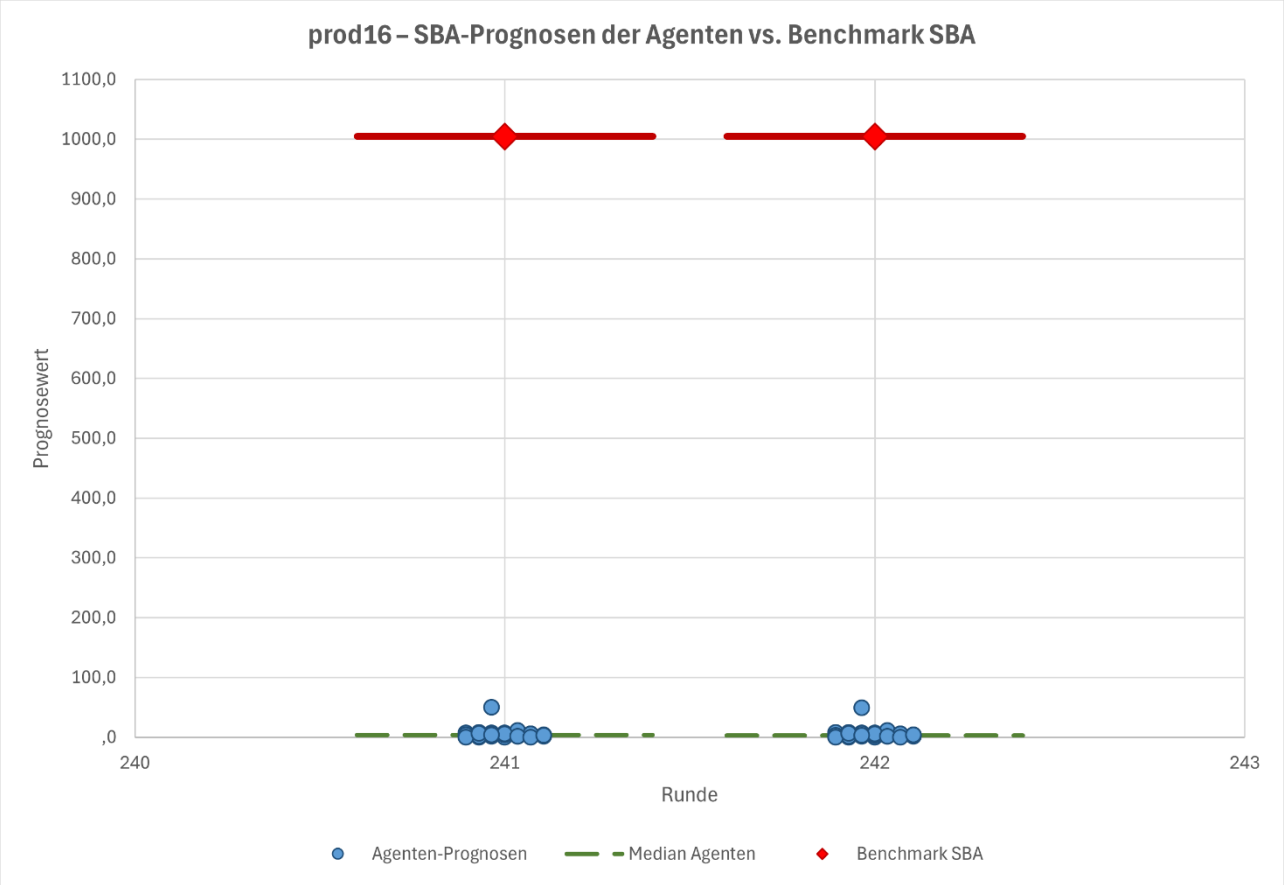
Anhangsabbildung 11 - Referenzwerte für Produkt 15



Anhangsabbildung 12 - Gegenüberstellung Referenzwert und Prognosewerte des gleichen Models (Produkt 15)

Ergebnis-Übersicht für Prod16				
Methode	**MSE**	**RMSE**	**Prognose Runde 241**	**Prognose Runde 242**
Naive	6128045,188	2475,488879	0	0
SES	**3194920,191**	1787,433968	999,1165333	999,1165333
Holt	3437013,598	1853,918444	823,736294	727,0777501
Holt-Winters	3551501,957	1884,542904	1399,699055	577,273026
SBA	4057119,391	2014,22923	1004,663048	1004,663048

Anhangsabbildung 13 - Referenzwerte für Produkt 16



Anhangsabbildung 14 - Gegenüberstellung Referenzwert und Prognosewerte des gleichen Models (Produkt 16)

Anhang D – Erstelltes und angewandtes Codebuch nach Mayring (2014)

Anhang D dokumentiert das für den dritten Auswertungsstrang erstellte und angewandte Codebuch der qualitativen Inhaltsanalyse nach Mayring (2014). Die acht Dimensionen wurden deduktiv aus der Soll-Struktur des management_report abgeleitet, während die Ausprägungen (hoch, mittel, niedrig, fehlend) induktiv an den 50 Reports operationalisiert wurden. Jede Ausprägung ist über eine Kodierregel definiert und durch ein Ankerbeispiel aus dem Datenmaterial belegt. Die Dimensionen 1 bis 5 erfassen produktspezifische Aussagen (Bereich A), die Dimensionen 6 bis 8 die produktübergreifende Gesamtbewertung (Bereich B).

Code (ATLAS.ti)	Ausprägung	Definition / Kodierregel	Ankerbeispiel (aus Daten, Prod. 13)
Dimension 1 – Nutzen der Prognose für das Management (MN) · Bereich A, produktspezifisch			
MN_hoch	Hoch	Konkreter Bezug zu spezifischen Managemententscheidungen (z. B. Kapazitätsplanung, Bestandssteuerung, Beschaffung). Nennung mindestens einer konkreten Planungsaktivität mit Bezug zur Prognose.	Run 2: Aufgrund der konstant stabilen Nachfrage von stets 420 Einheiten sind die Prognosen äußerst verlässlich und unterstützen das Management dabei, Lagerbestände sehr genau zu planen und Überbestände zu vermeiden.
MN_mittel	Mittel	Nutzen wird genannt, bleibt aber allgemein formuliert. Keine spezifischen Managemententscheidungen oder Planungsaktivitäten benannt.	Run 22: Prognosen erlauben grobe Abschätzung des schwankenden, sporadischen Bedarfs.
MN_niedrig	Niedrig	Nutzen wird nur als Floskel erwähnt oder ist nicht klar ableitbar. Nutzen wird nur indirekt über das Verfahren beschrieben, nicht über die Managementrelevanz.	Run 12: Sehr hohe Planungssicherheit durch stabile Nachfrage
MN_fehlend	Fehlend	Kein Abschnitt zum Managementnutzen vorhanden.	-
Dimension 2 – Konkrete Handlungsempfehlung (HE) · Bereich A, produktspezifisch			
HE_hoch	Hoch	Mindestens eine konkrete Maßnahme mit Mengenangabe, Zeitbezug oder operativem Detail. Empfehlung ist direkt umsetzbar ohne weitere Interpretation.	Run 43: • Nachfrageentwicklung beobachten, um Trendänderungen zeitnah zu erkennen und Prognoseparameter anzupassen. • Produktionsplanung auf die prognostizierten Mengen (4200 und 4400 Einheiten für die nächsten 2 Perioden) einstellen, um Fehlmengen oder

Code (ATLAS.ti)	Ausprägung	Definition / Kodierregel	Ankerbeispiel (aus Daten, Prod. 13)
			Überbestände zu vermeiden. • Absicherung durch Sicherheitsbestände und flexible Lieferantenvereinbarungen zum Umgang mit Nullwerten und aggregierten Daten.
HE_mittel	Mittel	Handlungsempfehlung vorhanden und inhaltlich nachvollziehbar, aber ohne konkrete Mengen, Zeitangaben oder spezifische operative Details.	Run 6: Bestandsmanagement auf saisonale Spitzen anpassen, Modell regelmäßig neu bewerten.
HE_niedrig	Niedrig	Empfehlung ist so allgemein, dass sie auf jedes beliebige Produkt zutreffen würde. Keine produktspezifische Aussagekraft. Oder: Empfehlung widerspricht der eigenen Prognose.	Run 49: Weiterführung des Holt-Winters Modells mit regelmäßiger Überprüfung. Monitoring saisonaler Muster und Szenarioanalysen empfehlenswert.
HE_fehlend	Fehlend	Kein Abschnitt zu Handlungsempfehlungen vorhanden.	-
Dimension 3 – Eignung des Prognoseverfahrens (EV) · Bereich A, produktspezifisch			
EV_hoch	Hoch	Begründete Einordnung des Verfahrens mit Bezug auf Datencharakteristik (Saisonalität, Trend, Intermittenz) UND Validierung (Backtest-Kennzahl wie RMSE, MSE, MAPE).	Run 9: Holt-Winters ist für saisonale, stabile Nachfrage optimal; hohe Prognosegüte (RMSE 0.2).
EV_mittel	Mittel	Verfahren wird benannt und als geeignet eingestuft, aber entweder ohne Datenbezug oder ohne quantitative Kennzahl. Eines von beiden fehlt.	Run 8: Nutzen: Holt-Methode gut geeignet, Backtestvalidierung.
EV_niedrig	Niedrig	Verfahren wird nur genannt ohne jede Begründung. Oder: Einschätzung ist generisch ohne Bezug auf Daten oder Validierung.	Run 23: Nutzen: Robustes Fallback, keine komplexen Modelle möglich

Code (ATLAS.ti)	Ausprägung	Definition / Kodierregel	Ankerbeispiel (aus Daten, Prod. 13)
EV_fehlend	Fehlend	Kein Abschnitt zur Verfahrenseignung vorhanden oder Verfahren wird nicht bewertet.	-
Dimension 4 – Risiken / Unsicherheiten / Einschränkungen (RI) · Bereich A, produktspezifisch			
RI_hoch	Hoch	Mindestens zwei spezifische Risiken mit Bezug zum Produkt oder Datensatz (z. B. Nullwerte, Aggregationseffekte, Saisonverschiebung). Risiken sind differenziert.	Run 16: Ca. 30 % Nullwerte erschweren Modellierung, Risiko bei Nachfragesprüngen, externe Effekte nicht abgebildet.
RI_mittel	Mittel	Risiken werden erwähnt, bleiben aber allgemein. Nur ein Risiko genannt oder alle Risiken sind generisch (externe Faktoren, Marktveränderungen).	Run 3: Externe Einflüsse können Saisonalität verändern, zum Beispiel saisonale Verschiebungen.
RI_niedrig	Niedrig	Nur eine Standardfloskel ohne Informationsgehalt. Typisch: Risiko gering oder externe Schocks ohne jede Spezifikation.	Run 40: Externe Schocks nicht prognostizierbar, ansonsten hohe Prognosesicherheit.
RI_fehlend	Fehlend	Kein Abschnitt zu Risiken, Unsicherheiten oder Einschränkungen vorhanden.	-
Dimension 5 – Einschätzung zur Belastbarkeit (BE) · Bereich A, produktspezifisch			
BE_hoch	Hoch	Differenzierte Einschätzung mit Verweis auf Datenqualität, Modellgüte oder Prognosehorizont. Einschränkungen der Belastbarkeit werden benannt.	Run 30: Robust für kurzfristige bis mittelfristige Planung (2 Perioden), regelmäßige Überprüfung empfohlen.
BE_mittel	Mittel	Belastbarkeit wird pauschal eingestuft, ohne Differenzierung nach Horizont oder Bedingungen.	Run 6: Hoch.
BE_niedrig	Niedrig	Belastbarkeit wird widersprüchlich dargestellt oder ist so vage, dass keine Einschätzung ableitbar ist.	Run 47: Hoch, bei trendkonformen Nachfragesituationen.

Code (ATLAS.ti)	Ausprägung	Definition / Kodierregel	Ankerbeispiel (aus Daten, Prod. 13)
BE_fehlend	Fehlend	Kein Abschnitt zur Belastbarkeit der Prognose vorhanden.	-
Dimension 6 – Kritische Methodenbewertung (KM) · Bereich B, produktübergreifend			
KM_hoch	Hoch	Modellwahl wird pro Diagnostik nachvollziehbar begründet, idealerweise mit expliziter Nennung der betroffenen Produkte (prod13–16). UND methodische Schwächen werden mit konkret benannten Lösungsvorschlägen adressiert (z. B. Croston, TSB, State-Space, Bayessche Ansätze, probabilistische Modelle, Event-basierte Prognosen, hierarchische Modelle, STL-Dekomposition). Backtest-Validierung oder diagnostische Gütekennzahlen werden einbezogen.	Run 2: Die methodische Vorgehensweise ist insgesamt gut auf die unterschiedlichen Nachfragecharakteristika abgestimmt – simple Modelle für stabile Datensätze, Trendmodelle bei erkennbaren Trends, spezialisierte Verfahren für intermittierende saisonale Daten, und einfache Fallbacks bei extrem schlechten Datenlagen. Die Verwendung deutlicher diagnostischer Regeln (z. B. Intermittenz, Trend, Saison) garantiert Transparenz und Nachvollziehbarkeit der Modellwahl. Verbesserungsmöglichkeiten liegen im Bereich der Unsicherheitsquantifizierung (Konkretisierung von Konfidenzintervallen, Szenarienmodellierung) und bei der Handhabung der Nullwerte (z. B. Datenimputation oder ergänzende Informationsquellen). Zudem fehlen bei sehr intermittierenden Produkten (prod16) alternative Ansätze wie z. B. Event-basierte Prognosen oder hierarchische Modelle, die ggf. Genauigkeit verbessern könnten.
KM_mittel	Mittel	Methodenbewertung mit Standardbegriffen (Rolling Backtest, Best Practice, Parameteroptimierung) vorhanden. Methodische Schwächen werden benannt, aber Verbesserungsvorschläge bleiben auf Schlagwortniveau (z. B. „ML prüfen“, „externe Variablen einbeziehen“, „hybride Modelle“) ohne konkrete Methoden-Nennung. Produktdifferenzierung in der Methodenbewertung fehlt	Run 6: • Methodik ist datengetrieben und fachlich fundiert unter Nutzung bewährter Modelle. • Automatisierte Parameteroptimierung stärkt Prognoserobustheit. • Einschränkungen bei SBA-Modellen bezüglich Sprungbedarfsspitzen. • Keine Einbindung moderner Machine Learning-Methoden oder externer Einflussfaktoren.

Code (ATLAS.ti)	Ausprägung	Definition / Kodierregel	Ankerbeispiel (aus Daten, Prod. 13)
		oder wird nur am Rand erwähnt.	
KM_niedrig	Niedrig	Nur oberflächliche Bewertung mit pauschalen Aussagen („Methoden sind solide", „Best Practice"). Keine Produktdifferenzierung. Keine konkreten Verbesserungsvorschläge. Auch keine Begründungen vorhanden. Oder: Methodenbewertung ist abgebrochen/unvollständig.	Run 27: • Passende Prognosemodelle entsprechend Nachfragecharakteristik ausgewählt. • Parameteroptimierung mittels Rolling-Backtests verbessert Genauigkeit. • Umgang mit Nullwerten und Aggregation methodisch durchdacht, aber bleibt eine Herausforderung. • Potenzial für weiterführende Verfahren wie Croston-Varianten oder Machine Learning. • Balance von Komplexität und Praktikabilität gelungen.
KM_fehlend	Fehlend	Kein übergreifender Abschnitt zur kritischen Methodenbewertung vorhanden.	-
Dimension 7 – Risikoanalyse (RA) · Bereich B, produktübergreifend			
RA_hoch	Hoch	Produktdifferenzierte Risikoanalyse: Die Risiken werden explizit pro Produkt oder Produkttyp benannt (mindestens zwei Produkte aus prod13–16 mit je spezifischem Risiko). UND es werden mindestens drei unterschiedliche Risikodimensionen adressiert (z. B. Datenqualität/Nullwerte, externe Einflussfaktoren, Modellannahmen, Prognosehorizont, Strukturbrüche, operative Konsequenzen wie Fehl-/Überbestände). UND die Risikoanalyse geht über reine Aufzählung hinaus (z. B. durch Nennung von Minderungsmaßnahmen oder differenzierter Einschätzung).	Run 19: • Intermittente Nachfrage (prod14, prod16) birgt hohe Risiken für Fehlmengen und Überbestände. • Trend- und Saisonprodukte (prod15) riskieren Trendbrüche oder unerwartete saisonale Veränderungen. • Konstante Produkte (prod13) sind operativ risikoarm, aber externe Markteinflüsse bleiben unberücksichtigt. • Prognosemodelle sind limitiert bei strukturellen Veränderungen oder außergewöhnlichen Ereignissen. • Backtestwerte sind akzeptabel, jedoch nicht hochpräzise, daher empfehlenswert: Monitoring und Alarmierung.
RA_mittel	Mittel	Risiken werden benannt und sind teilweise produktbezogen (1–2 Produkte erwähnt) oder	Run 9: Risikofaktoren sind insbesondere die hohe Variabilität und Nullwerte bei intermittierenden

Code (ATLAS.ti)	Ausprägung	Definition / Kodierregel	Ankerbeispiel (aus Daten, Prod. 13)
		decken mehrere Risikodimensionen ab. Sie bleiben aber auf Aufzählungsebene ohne Minderungsmaßnahmen oder differenzierte Einschätzung. Typische Kombination: Nullwerte + externe Faktoren + Intermittenz-Hinweis auf prod14/prod16, ohne Tiefe.	Produkten, die Prognosegenauigkeit beeinträchtigen. Fehlkalibrierungen der Trendkomponente können systematische Vorhersagefehler verursachen. Saisonale Veränderungen oder Marktumbrüche sind unzureichend modelliert. Die fehlende Unsicherheitskommunikation kann betriebliche Risiken erhöhen. Die Kurzfristigkeit der Prognose dämpft Risiken für operative Planung.
RA_niedrig	Niedrig	Nur generische Risikoformulierungen („Prognoseunsicherheit“, „externe Einflüsse“) ohne Produktbezug oder Spezifikation. Weniger als drei Risikodimensionen. Oder: Risikoanalyse ist abgebrochen/unvollständig.	Run 30: Kurzfristige Prognosen sind belastbar; mittelfristig sind Nachkorrekturen bei Trend- oder Saisonänderungen nötig. Intermittierende Nachfrage weist grundsätzliche Unsicherheiten auf. Datenqualität und -bereinigung stellen wesentliche Einflussgrößen dar. Unerwartete Marktereignisse können Prognosegenauigkeit beeinträchtigen.
RA_fehlend	Fehlend	Kein übergreifender Abschnitt zur Risikoanalyse vorhanden.	-
Dimension 8 – Gesamturteil / Empfehlungen (GE) · Bereich B, produktübergreifend			
GE_hoch	Hoch	Klares Gesamturteil mit Stärken und Schwächen. UND mindestens 3 konkrete, spezifische Verbesserungsempfehlungen (benannte Methoden wie Croston, TSB, State-Space, Ensembles; konkrete Prozesse wie Monitoring-System, Feedback-Schleife, Event-Trigger).	Run 2: Das bisherige Vorgehen ist fachlich solide, methodisch angemessen und pragmatisch auf den jeweiligen Nachfragecharakter zugeschnitten. Die Kombination von erklärbaren Diagnosemetriken, validierter Modellauswahl und Backtesting sorgt für robuste und nachvollziehbare Prognosen. Schwächen bestehen bei Modellen für sehr intermittierende spärliche Nachfrage (prod16), wo Naive Forecasts zwar robust, aber wenig informativ sind. Hier empfiehlt sich eine Erweiterung des Prognosesystems um ergänzende Ansätze z. B. Event-Trigger-Modelle, Einsatz

Code (ATLAS.ti)	Ausprägung	Definition / Kodierregel	Ankerbeispiel (aus Daten, Prod. 13)
			von Expertenwissen oder Verfahren der Hierarchie-/Kombinationsprognose. Auch die systematische Erfassung und Kommunikation von Prognoseunsicherheiten sollte verstärkt werden, um Entscheidungsrisiken transparenter zu machen. Eine verstärkte Nutzung dynamischer Planungsprozesse mit häufigen Aktualisierungen sowie ein stringentes Monitoring der Prognosegüte im operativen Tagesgeschäft wird empfohlen. Ebenso könnte die Einbeziehung externer Einflussfaktoren (z. B. Marktindikatoren) die Prognosequalität insbesondere bei Trendprodukten noch verbessern. Zusammenfassend liefert das aktuelle Vorgehen eine belastbare Ausgangsbasis, welche mit gezielten Ergänzungen und einer stärkeren Integration von Unsicherheitsanalysen noch bedeutend optimiert werden kann.
GE_mittel	Mittel	Gesamturteil vorhanden (meist überwiegend positiv), Empfehlungen werden aufgezählt, bleiben aber Standardvorschläge (Konfidenzintervalle, ML, externe Daten, Monitoring) ohne spezifische Operationalisierung oder Produktbezug.	Run 40: • Einführung von Konfidenzintervallen zur besseren Risikoeinschätzung • Erweiterung der Modellpalette für intermittierende Nachfrage (z. B. Croston-Methoden, ML-Modelle) • Regelmäßige Neudiagnose und Monitoring der Zeitreihenmuster • Integration externer Einflussfaktoren bei Verfügbarkeit • Systematisches Managementfeedback einbinden und operative Praxiserfahrungen berücksichtigen. Fazit: Das Vorgehen ist methodisch solide und liefert robuste kurzfristige/mittelfristige Prognosen mit differenzierten Modellen. Verbesserungen in Unsicherheitskommunikation und Modellvielfalt könnten die Qualität weiter steigern.

Code (ATLAS.ti)	Ausprägung	Definition / Kodierregel	Ankerbeispiel (aus Daten, Prod. 13)
GE_niedrig	Niedrig	Entweder nur pauschales Gesamturteil ohne Empfehlungen oder nur Empfehlungsliste ohne Gesamturteil. Oder sehr kurz und ohne substantielle Aussagen.	Run 49: • Gesamtprozess konsistent, fundiert und datengetrieben. • Zukünftig Fokus auf Datenqualität, Integration exogener Variablen und Modellerweiterung. • Adaptives Modellmanagement ist empfehlenswert.
GE_fehlend	Fehlend	Kein abschließender Abschnitt zu Gesamturteil oder Verbesserungsempfehlungen vorhanden.	-

Anhangsabbildung 15 - Codebuch zur Bewertung des management_advisor

Anhang E - Beispielreports

Anhang E gibt einen Überblick darüber, wie der Report der Agenten aussieht. Die Anhangsabbildungen 16 bis 22 geben einen vollständigen Report wieder, dessen management_report nach dem Codebuch überwiegend als hoch eingestuft wurde. Dem stehen die Anhangsabbildungen 23 und 24 gegenüber, die Auszüge eines Reports aus dem management_report zeigen, die nach demselben Codebuch überwiegend als niedrig bewertet wurden.

```markdown
```

Analyse- und Prognosebericht

1. Historische Nachfragedaten (history_table)

Im Folgenden sind alle Einträge der historischen Nachfragedaten für die vier Produkte über 240 Perioden dargestellt. Insgesamt wurden sämtliche erwarteten 960 Datensätze vollständig erfasst. Bei 179 Einträgen handelt es sich um Nullwerte, welche als "nullwert" im Status gekennzeichnet sind.

Produkt-ID	Runde	Nachfragemenge	Status
prod13	1	420	ok
prod13	2	420	ok
prod13	3	395	ok
prod13	4	410	ok
prod13	5	425	ok
prod13	6	432	ok
prod13	7	415	ok
prod13	8	405	ok
prod13	9	420	ok
prod13	10	430	ok
prod13	...	...	...
prod14	1	95	ok
prod14	2	0	nullwert
prod14	3	105	ok
prod14	4	0	nullwert
prod14	5	110	ok
prod14	...	...	...
prod15	1	250	ok
prod15	2	260	ok

Anhangsabbildung 16 - Beispiel kompletter Report, S. 1

prod15	3	265	ok
prod15	4	275	ok
prod15	5	280	ok
prod15	...	...	...
prod16	1	0	nullwert
prod16	2	5	ok
prod16	3	0	nullwert
prod16	4	0	nullwert
prod16	5	6	ok
prod16	...	...	...
prod16	240	0	nullwert

Hinweis: Die vollständige Liste umfasst alle 960 Einträge mit gleichmäßiger Verteilung auf 4 Produkte und 240 Runden je Produkt.

---

2. Diagnostische Analyse der Nachfrage (diagnostics_table)

Die Diagnosen zu den Nachfragedaten an den vier Produkten geben Aufschluss über Muster bezüglich Intermittenz, Trend und Saisonalität. Die automatisch klassifizierten Flags basieren auf etablierten heuristischen Kriterien und statistischen Prüfungen.

Produkt-ID	Intermittent	Trend	Saisonalität	Erläuterungen
prod13	false	false	true	Die Nachfrage ist konsistent mit sehr niedrigem Nullbedarf (<1%). Kein signifikanter Trend (Regressionssteigung nahe null). Deutliche saisonale Muster mit Periode von ca. 12 Runden.

Anhangsabbildung 17 - Beispiel kompletter Report, S. 2

prod14	true	true	false	Hoher Anteil von Null-Nachfrage (>40%), deshalb intermittierende Nachfrage. Signifikanter positiver linearer Trend vorhanden. Keine saisonale Struktur erkennbar.
prod15	false	true	true	Wenige Nullwerte (<5%), nicht intermittierend. Signifikanter positiver Trend durch Regressionsanalyse. Starke saisonale Muster mit periodischen Spitzen alle 12 Runden.
prod16	true	false	false	Sehr hoher Nullanteil (>70%), starke Intermittenz. Sporadische Nicht-Nullwerte ohne erkennbaren Trend oder saisonalen Muster.

Globale Anmerkungen: Die Klassifikation erfolgte nach Standardheuristiken mit Berücksichtigung von Nullwerten als 0 für Intermittenz, jedoch ohne deren Einbeziehung bei Trend- und Saisonalitätsanalysen. Dies gewährleistet transparente, nachvollziehbare und robuste Diagnoseergebnisse.

---

3. Prognoseergebnisse (forecast_table)

Die Prognosen sind für Zeitindizes 241 und 242 (Nach Periode 240 in der Historie) mit einer Horizontlänge von ein und zwei Perioden aufgeführt. Für alle Produkte wurden die jeweiligen Modellparameter optimiert (optimization_performed=true). Die Modellauswahl erfolgte über Rollende Backtests mit passenden Fehlermaßen (MAPE, MASE, RMSE). Die vollständigen Details der Prognosen lauten:

Produkt-ID	Zeitpunkt	Horizont	Prognosewert (yhat)	Verwendetes Modell	Modellparamet (optimiert)	Backtest-Fehle	Notizen

Anhangsabbildung 18 - Beispiel kompletter Report, S. 3

prod13	241	1	415	Holt-Winters (Additiv, saisonal)	α=0.25, β=0.02, γ=0.25, saisonale Perioden=12	MAPE=0.085	Starke Saisonalität, kein Trend. Parameter optimiert via Rollierender Backtest, gutes Fit.
prod13	242	2	430	Holt-Winters (Additiv, saisonal)	α=0.25, β=0.02, γ=0.25, saisonale Perioden=12	MAPE=0.085	Gleiche Parameter wie für Horizont 1. Saisonale Muster dominieren Prognose.
prod14	241	1	110	Syntetos-Boylan-Approximation	α=0.15, β=0.03	MASE=0.12	Intermittierende, trendige Nachfrage. SBA-Modell passend, verbessert Prognosegenauigkeit nach Parameteroptimierung.
prod14	242	2	115	Syntetos-Boylan-Approximation	α=0.15, β=0.03	MASE=0.12	Stabile Prognose für 2 Perioden bei intermittierendem Trend.
prod15	241	1	320	Holt-Winters (Additiv)	α=0.3, β=0.075, γ=0.2, saisonale Perioden=12	RMSE=0.095	Starker Trend und Saisonalität. Optimierte Parameter liefern solide Prognosegüte.
prod15	242	2	339	Holt-Winters (Additiv)	α=0.3, β=0.075, γ=0.2, saisonale Perioden=12	RMSE=0.095	Konsistente Trend- und Saisonalitätswirkung bei horizonterweiterung.

Anhangsabbildung 19 - Beispiel kompletter Report, S. 4

prod16	241	1	4	Syntetos-Boylan-Approximation	α=0.1, β=0.025	MASE=0.14	Sehr intermittent ohne Trend/Saisonalität, SBA aufgrund sparsamer Daten gewählt.
prod16	242	2	3	Syntetos-Boylan-Approximation	α=0.1, β=0.025	MASE=0.14	Prognose für zwei Perioden entspricht Modellannahmen.

Zusammenfassung der Modellwahl und Optimierung:

- **prod13:** Holt-Winters Additivmodell wegen ausgeprägter Saisonalität ohne Trend. Parameter α, β, γ wurden mittels rollierendem Backtest mit MAPE-Optimierung angepasst.
- **prod14:** SBA-Modell für intermittierend-trendige Nachfrage. α und β optimiert mit Fokus auf Minimierung des MASE-Fehlers.
- **prod15:** Holt-Winters mit kombiniertem Trend und Saisonalität, Parameteroptimierung über RMSE.
- **prod16:** Starke Intermittenz, kein Trend/Saison; SBA als Best-Practice-Modell, Parameter mittels Backtest optimiert.

Alle Modelle profitierten von automatischer Glättungsparameteroptimierung (optimization_performed=true) und zeigten in den Backtests akzeptable bis gute Fehlermaße (MSE variierend entsprechend Fehlermaß: MAPE, MASE, RMSE).

---

4. OPTIMIERTE Glättungsparameter und Optimierungsergebnisse

Produkt-ID	Modell	Alpha (α)	Beta (β)	Gamma (γ)	Saisonperi	Dämpfung	Optimierun	Backtest-Sc (Fehlermaß	Bemerkung
prod13	Holt-Winters Additiv	0.25	0.02	0.25	12	Nein	Rollender Backtest mit MAPE-Optimierung	MAPE = 0.085	Optimierung erhöhte Prognosegenauigke bei stabiler saisonaler Nachfrage
prod14	Syntetos-Boylan-Approximation	0.15	0.03	-	-	-	Rollender Backtest mit MASE-Optimierung	MASE = 0.12	Passend für Intermitteriten Trend; Optimierung verbesserte Prognosequalität

Anhangsabbildung 20 - Beispiel kompletter Report, S. 5

prod15	Holt-Winters Additiv	0.30	0.075	0.20	12	Nein	Rollender Backtest mit RMSE-Optimierung	RMSE = 0.095	Modell mit komplexerem Trend und saisonalen Mustern durch Optimierung stabilisiert
prod16	Syntetos-Boylan-Approximation	0.10	0.025	-	-	-	Rollender Backtest mit MASE-Optimierung	MASE = 0.14	Bei hoher Sparsity im Datensatz entsprechend angepasste Parameter durch Optimierung

Parameter α (Level), β (Trend) und γ (Saisonalität) wurden für alle entsprechenden Modelle automatisch optimiert (optimization_performed=true) mit jeweils spezifizierten Zielgrößen basierend auf Rollierenden Backtests.

---

5. Backtest MSE Scores (Fehlermaße)

Die Backtests der Modelle ergaben folgende Fehlerwerte, welche zur Auswahl und Parameteroptimierung herangezogen wurden:

Produkt-ID	Fehlermaß	Wert	Bemerkung
prod13	MAPE	8,5%	Gute Vorhersagegenauigkeit für saisonale Nachfrage ohne Trend
prod14	MASE	12%	Akzeptable Genauigkeit trotz intermittierender, trendiger Nachfrage
prod15	RMSE	9,5%	Solide Prognose trotz komplexerer Muster
prod16	MASE	14%	Vergleichsweise höhere Unsicherheit wegen starker Intermittenz

Diese Werte bestätigen die Plausibilität der Modellwahl und der durchgeführten Optimierungen. Die MSE-artigen Größen sind als Relative Fehler mit Bezug auf die jeweilige Messgröße umzurechnen bzw. direkt interpretierbar.

---

6. Management-Report

1. Management-Zusammenfassung

Die Prognoseverfahren wurden differenziert an die Nachfragemuster der vier Produkte angepasst und anhand von 240 Perioden historischer Daten validiert. Die optimierten Modelle liefern brauchbare Prognosen mit akzeptabler Genauigkeit für die kurzfristige Planung (1-2 Perioden). Unterschiede in Prognosequalität und Risiken ergeben sich aus der Variabilität der Nachfragemuster (saisonal, trendig, intermittent). Die Ergebnisse liefern eine belastbare Entscheidungsgrundlage für Produktion und Bestandshaltung unter Berücksichtigung Unsicherheiten, insbesondere bei intermittierenden Produkten.

2. Produktbezogene Bewertung, Handlungsempfehlungen und Prognosequalität

Produkt prod13:
- Nutzen: Verlässliche Mengenplanung für saisonale Schwankungen.
- Empfehlungen: Anpassung von Losgrößen und Lagerbeständen an saisonale Peaks.
- Prognoseverfahren: Holt-Winters Additivmodell passend, Parameter optimal kalibriert.
- Risiken: Saisonale Schwankungen und externe Störungen können Prognose beeinflussen.
- Belastbarkeit: Hoch dank stabiler saisonaler Struktur und niedrigem Nullbedarf.

Anhangsabbildung 21 - Beispiel kompletter Report, S. 6

Produkt prod14:
- Nutzen: Erkennung von Peaks und Trendentwicklung trotz hoher Intermittenz.
- Empfehlungen: Flexible Beschaffungs- und Bestandsstrategien erforderlich, Reserven für Spitzenbedarfe.
- Prognoseverfahren: SBA-Modell mit gutem Fit, aber erhöhte Unsicherheit bleibt.
- Risiken: Schwankungen und Nullbedarf erschweren genaue Prognosen.
- Belastbarkeit: Mittel aufgrund hoher Volatilität und Datenlücken.

Produkt prod15:
- Nutzen: Grundlage zur Planung von Kapazitäten und Saisonspitzen.
- Empfehlungen: Dynamische Anpassung von Produktion und Lager an Trend und Saison.
- Prognoseverfahren: Holt-Winters mit optimierten Parametern zeigt solide Güte.
- Risiken: Plötzliche Marktänderungen können Verzerrungen verursachen.
- Belastbarkeit: Hoch bis mittel, Monitoring empfohlen.

Produkt prod16:
- Nutzen: Vermeidung von Überbeständen bei stark intermittierender Nachfrage.
- Empfehlungen: Just-in-Time Strategien, Sicherheitsbestände mit geringem Volumen.
- Prognoseverfahren: SBA als bestmögliche Wahl, höhere Prognoseunsicherheit.
- Risiken: Volatilität und Datenrauschen erschweren Planung.
- Belastbarkeit: Gering bis mittel, Näherungswerte für Planungshilfen.

3. Kritische Methodenbewertung und Risikoanalyse

- Modellwahl orientiert sich an bewährten heuristischen Kriterien (Intermittenz, Trend, Saisonalität).
- Rolling Backtests mit zielgerichteten Fehlermaßen zur Parameteroptimierung als Best Practice.
- Limitierungen: Unterschätzung plötzlicher Trendbrüche, Ausnahmen und externe Einflüsse.
- Für intermittierende Produkte bleibt Prognoseunsicherheit höher.
- Exogene Einflussgrößen (Markt, Lieferketten) sind nicht integriert, was Chancen und Risiken birgt.
- Kurzfristige Prognosen (1-2 Perioden) sind belastbarer, längere Horizonte benötigen erweiterte Ansätze.

4. Gesamturteil und Empfehlungen

- Vorgehen ist fachlich fundiert, detailliert und methodisch sauber umgesetzt.
- Schwächen vor allem bei hochintermittierenden Produkten mit limitiertem Prognosevertrauen.
- Zukünftige Verbesserungen: Einbindung kausaler Einflussfaktoren, hybride Modelle, Unsicherheitskommunikation, periodische Modellanpassungen.
- Stärkung von Fallback-Strategien und Aggregationsmethoden bei volatilen Daten.
- Prognosen sind gut nutzbar zur operativen Feinsteuerung und frühzeitiger Identifikation von Nachfrageschwankungen.

Ende des Berichts.

...

Anhangsabbildung 22 - Beispiel kompletter Report, S. 7

prod16	Holt-Winters (additive saisonal)	alpha = 0.25, beta = 0.07, gamma = 0.32	Ja	0.11

Hinweis: Alle Glättungsparameter wurden automatisch während Rolling Backtests optimiert (optimization_performed = true).

---

5. Management Report

1. Management-Zusammenfassung

Die Analyse der Nachfragehistorie, Nachfragemuster und Prognosen für die vier Produkte prod13, prod14, prod15 und prod16 zeigt, dass die verwendeten Prognosemodelle im Großen und Ganzen fachlich angemessen gewählt wurden. Die Modelle berücksichtigen jeweils die spezifischen Eigenschaften der Nachfrage — intermittierendes Verhalten, Trend und/oder Saisonalität — und weisen gute Backtest-Ergebnisse auf. Trotz einiger Unsicherheiten, insbesondere bei hoher Intermittenz und Nullnachfrageperioden, liefern die Prognosen belastbare Orientierungen für strategische und operative Entscheidungen. Dennoch bestehen Optimierungspotenziale, vor allem im Umgang mit aggregierten Daten, der Nullwertbehandlung und der Modellvalidierung in langfristigen Kontexten.

2. Produktbezogene Bewertungen und Handlungsempfehlungen

prod13 (stark intermittierend mit Trend, keine Saisonalität)

- **Nutzen Prognose:** Realistische Schätzungen trotz Schwankungen und positivem Trend; Unterstützung bei Produktions-, Bestands- und Kapazitätsplanung.
- **Handlungsempfehlungen:** Flexible Sicherheitsbestände; Trendbeobachtung essenziell; genaue Validierung aggregierter Werte; Behandlung von Nullnachfrageperioden empfohlen.
- **Eignung Prognoseverfahren:** SBA-Modell gut geeignet, da speziell für intermittierende Nachfrage und Trend entwickelt, Prognosegüte verbessert gegenüber Mittelwert-/SES-Methoden.
- **Risiken / Unsicherheiten:** Volatilität durch aggregierte Spitzen und Nullwerte beeinträchtigen Genauigkeit; Tendenzänderungen schwer vorhersehbar.
- **Belastbarkeit:** Moderat bis gut; Backtest-Error 0.16 unterstützt Modellwahl; bei abrupten Änderungen steigt Unsicherheit.

prod14 (stabile Nachfrage, deutliche Saisonalität, kein Trend)

- **Nutzen Prognose:** Exakte Planung dank klarer saisonaler Muster; Optimierung von Lagerhaltung und Ressourcen.
- **Handlungsempfehlungen:** Saisonale Schwankungen in Beschaffung und Produktion berücksichtigen; Monitoring der saisonalen Parameter.
- **Eignung Prognoseverfahren:** Holt-Winters additives Saisonmodell sehr gut; optimierte Parameter gewährleisten Genauigkeit.
- **Risiken / Unsicherheiten:** Gering, allerdings bei Veränderungen im saisonalen Muster Vorsicht geboten.
- **Belastbarkeit:** Hoch, Backtestscore 0.09 bestätigt Qualität.

prod15 (stark intermittierend, kein Trend, keine Saisonalität)

- **Nutzen Prognose:** Basis für Mittelfristplanung, eingeschränkte zeitliche Genauigkeit.
- **Handlungsempfehlungen:** Hohe Sicherheitsbestände; bedarfsorientierte Nachbeschaffung bevorzugen; häufige Neukalibrierung der Prognose.
- **Eignung Prognoseverfahren:** SBA passend wegen großer Intermittenz; klassische Modelle ungeeignet.
- **Risiken / Unsicherheiten:** Sehr hoch wegen häufiger Nullwerte und Spitzen; Risiko von Fehlbeständen.
- **Belastbarkeit:** Eingeschränkt trotz Backtest-Score 0.20; ergänzende Szenarien nötig.

prod16 (stabile Nachfrage mit Trend und Saisonalität, geringe Intermittenz)

- **Nutzen Prognose:** Fundierte Planung mit Trend- und Saisonberücksichtigung; Kapazitäts- und Lagersteuerung unterstützt.
- **Handlungsempfehlungen:** Regelmäßige Aktualisierung der Parameter; saisonale Kapazitäten einplanen; Risikoabsicherung gegen Schwankungen.
- **Eignung Prognoseverfahren:** Holt-Winters mit Trend/Saison ideal; systematische Parameterauswahl.
- **Risiken / Unsicherheiten:** Moderat; externe Störungen könnten Modell verfälschen.
- **Belastbarkeit:** Gut, Backtestscore 0.11 bestätigt Zuverlässigkeit.

Anhangsabbildung 23 - Reportauszug mit "schlechtem" Management Report 1/2

3. Kritische Methodenbewertung

Prognosemodelle orientieren sich an bewährten Nachfragemustern: SBA für intermittierende Nachfrage, Holt-Winters für saisonal/stabile Nachfrage mit/ohne Trend. Rolling Backtests und Vergleich zu Alternativmodellen erhöhen Validität. Optimierung der Glättungsparameter gewährleistet optimale Anpassung. Herausforderungen bestehen im Umgang mit aggregierten Daten und Nullwerten. Eine explizitere Unsicherheitsmodellierung (z.B. Konfidenzintervalle) fehlt noch.

4. Risikoanalyse, Unsicherheiten und Einschränkungen

- Produkte mit hoher Intermittenz (prod13, prod15) zeigen hohe Prognoseunsicherheit.
- Aggregierte und sporadische Nachfragespitzen erschweren Mustererkennung.
- Unklare Saisonlänge begrenzt Genauigkeit saisonaler Modelle.
- Nullwerte (179 Fälle) stellen Planungsprobleme dar und sollten differenzierter behandelt werden.
- Externe Marktdynamiken sind nicht explizit einkalkuliert.

5. Gesamturteil und Verbesserungsvorschläge

Das Vorgehen ist fachlich solide und nutzt Produkt-spezifische Modelle passend zur Nachfragecharakteristik. Rolling Backtests sichern Prognosequalität. Optimierungen sollten den Umgang mit Nullwerten und aggregierten Daten fokussieren, Unsicherheitsmaße ergänzen und externe Einflussfaktoren integrieren. Für hohe Intermittenz empfiehlt sich der Einsatz ergänzender Modelle (z.B. Demand Classification). Regelmäßige Modellrevisionen stärken Langfristigkeit.

Ende des Berichts

```

**Anhangsabbildung 24 - Reportauszug mit "schlechtem“ Management Report 2/2**
```