# Projekt: LLM-gesteuerte Analyse von Maschinendaten

**Eine iterative Reise zur Validierung einer "Zero-Algorithm"-Strategie**

## Die Herausforderung & Die Vision

### **Die Kernfrage (Die Herausforderung)**

Können wir Maschinendaten ausschließlich mit einem Large Language Model (LLM) analysieren, ohne vordefinierte Algorithmen zu verwenden?

cite: 01-scope.md

* **Ziel:** Direkte Beantwortung von Fragen in natürlicher Sprache (z.B. "Was war der längste Zyklus?").
* **Wichtige Anforderung:** Der Ansatz muss universell sein und darf keine hartcodierte Geschäftslogik enthalten.

### **Die Strategische Vision**

Die Forschung von 2024-2025 bestätigt: Der Ansatz ist mit modernen Technologien hochgradig machbar.

cite: technical-research-analysis.md

* **Empfohlene Architektur:** Eine Kombination aus Zeitreihen-spezialisierten LLMs (Time-LLM) und Retrieval-Augmented Generation (RAG).
* **Geschäftlicher Nutzen:** Enormes Potenzial durch Reduzierung manueller Analysen und schnellere, datengestützte Entscheidungen.

## Fünf Schritte zur Erkenntnis

Wir haben einen systematischen Ansatz verfolgt, um die Komplexität schrittweise zu steigern und aus jeder Phase zu lernen.

Notebook 1 (Der Anfang) ➔ Notebook 2 (Der Durchbruch) ➔ Notebook 3 (Die Nagelprobe) ➔ Notebook 4 (Der Beweis) ➔ Notebook 5 (Die Plattform)

## Iteration 1 – Der "Brute-Force"-Ansatz

* **Notebook:** mazak\_pure\_llm\_analysis.ipynb
* **Ansatz:** Ein sehr langer, strenger Prompt mit vielen Regeln ("ABSOLUTE RULES"), der das LLM zu korrektem Verhalten zwingen sollte. Die Kommunikation erfolgte über einfache HTTP-Anfragen.
* **Wichtige Funktionen:**
  + UltraFocusedLLMClient.analyze\_data
  + CriticalFixedQueryProcessor.process\_question
* **Gestellte Fragen:**
  + "Was war der längste Zyklus in den ACTIVE Daten?"
  + "Wie viele verschiedene Programme wurden im ACTIVE Modus ausgeführt?"
* **Ergebnis:**
  + **Genauigkeit: 🔴 25,0%**
  + **Problem:** Das Modell ignorierte die Regeln, halluzinierte Python-Code und gab widersprüchliche Antworten.
* **Erkenntnis:** Ein einfacher, strenger Befehl an ein kleines LLM reicht nicht aus. **Der Ansatz war nicht tragfähig.**

## Iteration 2 – Der Durchbruch mit LangChain

* **Notebook:** mazak\_pure\_langchain\_zero\_algorithm.ipynb
* **Ansatz:** Einführung des LangChain-Frameworks. Implementierung eines intelligenten **zweistufigen Prozesses**:
  1. **Verstehen:** Das LLM analysiert zuerst die Datenstruktur autonom.
  2. **Antworten:** Es beantwortet die Frage basierend auf seinem eigenen, zuvor erstellten Verständnis.
* **Wichtige Funktionen:**
  + PureLangChainAnalyzer.understand\_data\_universally
  + PureLangChainAnalyzer.answer\_question\_universally
* **Gestellte Fragen:**
  + "Was war der längste Zyklus in den ACTIVE Daten?"
* **Ergebnis:**
  + **Genauigkeit: 🟡 43,8%**
  + **Verbesserung:** Deutlich stabilere und relevantere Antworten.
* **Erkenntnis:** Die Strukturierung des "Denkprozesses" des LLM ist entscheidend für den Erfolg.

## Iteration 3 – A/B-Testing & Chain of Thought

* **Notebook:** phase1\_enhanced\_testing\_chain\_of\_thought.ipynb
* **Ansatz:** Einführung des **"Chain of Thought" (CoT)** Prinzips, um das Modell zum schrittweisen Denken zu zwingen, und A/B-Tests verschiedener Prompt-Strategien.
* **Wichtige Funktionen:**
  + EnhancedPureLangChainAnalyzer.answer\_question\_with\_chain\_of\_thought
* **Gestellte Fragen:**
  + "Was war der längste Zyklus?", "Wie viele verschiedene Programme wurden ausgeführt?" (mit strengerer Validierung)
* **Ergebnis:**
  + **Genauigkeit: 🟠 57,2%**
  + **Problem:** Die CoT-Idee ist gut, aber das kleine Modell llama3.2:1b ignoriert die komplexen Anweisungen (CoT-Nutzung: 0,0%).
* **Erkenntnis:** Die Qualität des Prompts ist wichtig, aber die Fähigkeiten des Modells sind der limitierende Faktor.

## Iteration 4 – Die Nagelprobe mit komplexen Fragen

* **Notebook:** phase2\_advanced\_continuation.ipynb
* **Ansatz:** Ein professionelles Validierungs-Framework, um die Ansätze ("Expert" vs. "Universal") mit wirklich **komplexen analytischen Fragen** zu testen.
* **Wichtige Funktionen:**
  + EnhancedValidationAlgorithms.validate\_llm\_response
  + WinnerSelectionSystem.determine\_winner
* **Gestellte Fragen (Beispiele):**
  + "Analysieren Sie die Korrelation zwischen Programmkomplexität und Zykluszeitvariationen."
  + "Vergleichen Sie die Ausführungseffizienz über verschiedene Betriebsmodi hinweg."
* **Ergebnis (Der Moment der Wahrheit):**
  + **Objektiv validierte Genauigkeit: 🔴 ~0%**
* **Erkenntnis:** Das Framework funktioniert perfekt und **beweist objektiv**, dass das llama3.2:1b-Modell für komplexe Analysen **ungeeignet** ist.

## Iteration 5 – Die produktionsreife Plattform

* **Notebook:** phase3\_fixed\_dependencies.ipynb
* **Ansatz:** Konsolidierung aller Erkenntnisse in einem stabilen, produktionsreifen System mit Fehleranalyse, Caching und Multi-Modell-Unterstützung.
* **Vergleich der Systemkomponenten:**

| **Merkmal** | **Alter Ansatz (Notebook 1)** | **Finaler Ansatz (Notebook 5)** |
| --- | --- | --- |
| **LLM-Kommunikation** | Einfache HTTP-Anfragen | Strukturiert über LangChain |
| **Fehlerbehandlung** | Manuell / Einfach | Systematische Fehleranalyse & Kategorisierung |
| **Performance** | Langsam, keine Optimierung | Intelligentes Memory-Caching |
| **Modell-Flexibilität** | Hartcodiert (llama3.2:1b) | Multi-Modell-Framework (Vergleich verschiedener Modelle) |
| **Validierung** | Einfache Referenz-Algorithmen | Umfassendes System mit statistischer Analyse |

* **Ergebnis:** Ein robustes Framework, bereit für den Einsatz mit leistungsstarken Modellen.
* **Erkenntnis:** Der Prototyp wurde erfolgreich in ein produktionsreifes System überführt.

## Zusammenfassung der Ergebnisse & Finale Erkenntnis

### Die Entwicklung der Genauigkeit

| **Notebook** | **Ansatz** | **Gemessene Genauigkeit** |
| --- | --- | --- |
| mazak\_pure\_llm\_analysis.ipynb | Brute-Force Prompt | 🔴 **25,0%** |
| mazak\_pure\_langchain\_zero\_algorithm.ipynb | 2-stufiger Prozess (LangChain) | 🟡 **43,8%** |
| phase1\_enhanced\_testing\_chain\_of\_thought.ipynb | Chain of Thought (CoT) | 🟠 **57,2%** |
| phase2\_advanced\_continuation.ipynb | Komplexe Fragen & Validierung | 🔴 **~0%** (objektiv validiert) |

### Finale Erkenntnis

Der "Pure LLM"-Ansatz ist technisch machbar und der entwickelte Rahmen ist extrem leistungsfähig. Der Erfolg ist jedoch **kritisch von der Leistungsfähigkeit des zugrundeliegenden LLM-Modells abhängig.** Das lokale Modell llama3.2:1b ist für komplexe, geschäftskritische Analysen unzureichend.

## Nächste Schritte & Empfehlung

### Empfehlung: 🟢 **PROCEED – Mit leistungsfähigeren Modellen fortfahren**

1. Leistungsstarke Modelle testen:  
   Das finale Framework (phase3\_fixed\_dependencies.ipynb) nutzen, um branchenführende Modelle zu testen (z.B. GPT-4o, Claude 3, größere Llama-Modelle).  
   cite: 02-decisions.md
2. Kosten-Nutzen-Analyse durchführen:  
   Mithilfe des Frameworks das Modell mit dem besten Verhältnis von Kosten, Geschwindigkeit und Genauigkeit für den produktiven Einsatz auswählen.  
   cite: 03-tasks.md, technical-research-analysis.md
3. Pilot-Implementierung:  
   Das beste Modell in einer Pilotphase auf einer einzelnen Maschine einsetzen, um den geschäftlichen Nutzen im realen Betrieb zu validieren.
4. Skalierung:  
   Nach erfolgreicher Pilotphase das System auf weitere Maschinen und Anwendungsfälle ausrollen.