Johnson & Johnson

Specification Is All You Need

Aditya Ujwal, Aidan Tomlin, Ben Espinoza, Henry Chen, Kush Kodiya, Neeil Gupta, Rakshit Pradhan, Rohit Karthik, Vasudha Koneru

PURDUE UNIVERSITY | The Data Mine

BACKGROUND

- Clinical data pipelines for survival analysis are complex, brittle, and hard to reproduce small changes to data formats or variables can break everything

- LLMs can generate code quickly but hallucinate and drift from specifications, making them unreliable for rule-heavy clinical data work without a structured approach

- Last semester's pipeline was built manually through iterative Jupyter notebook work, meaning it's tightly coupled to the Dream Pool dataset and can't easily generalize to new data or new clinical questions

GOALS

- Design specs and commands precise enough that the LLM output is consistently correct
- Generalize the pipeline so it adapts to different datasets and variable schemas rather than hardcoded to one dataset
- Build in automatic error recovery so the system retries and self-corrects without needing manual intervention
- Establish a reusable framework that other team members can extend with new spec version
- Test different LLM models to discover which models follow the spec the best and create the most optimal output.

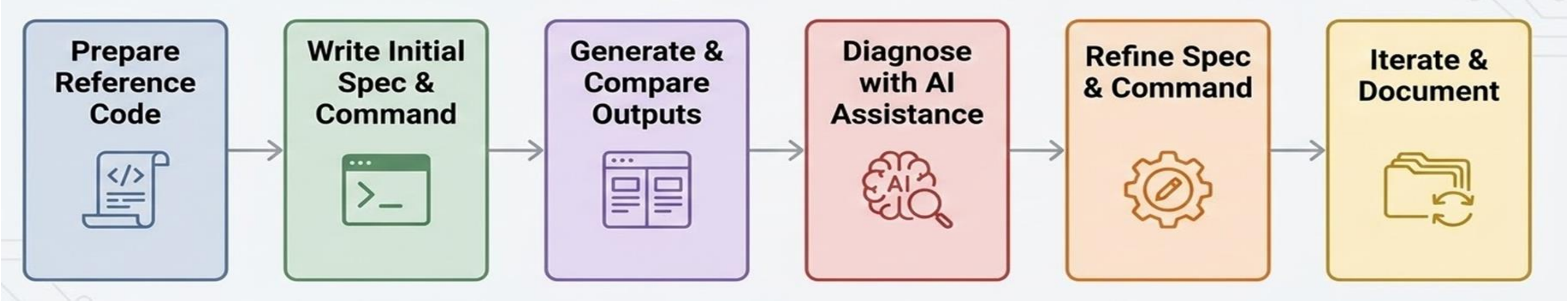
Spec → Hallucinates → LLM → Drifts from Spec → Output

Claude

Gemini

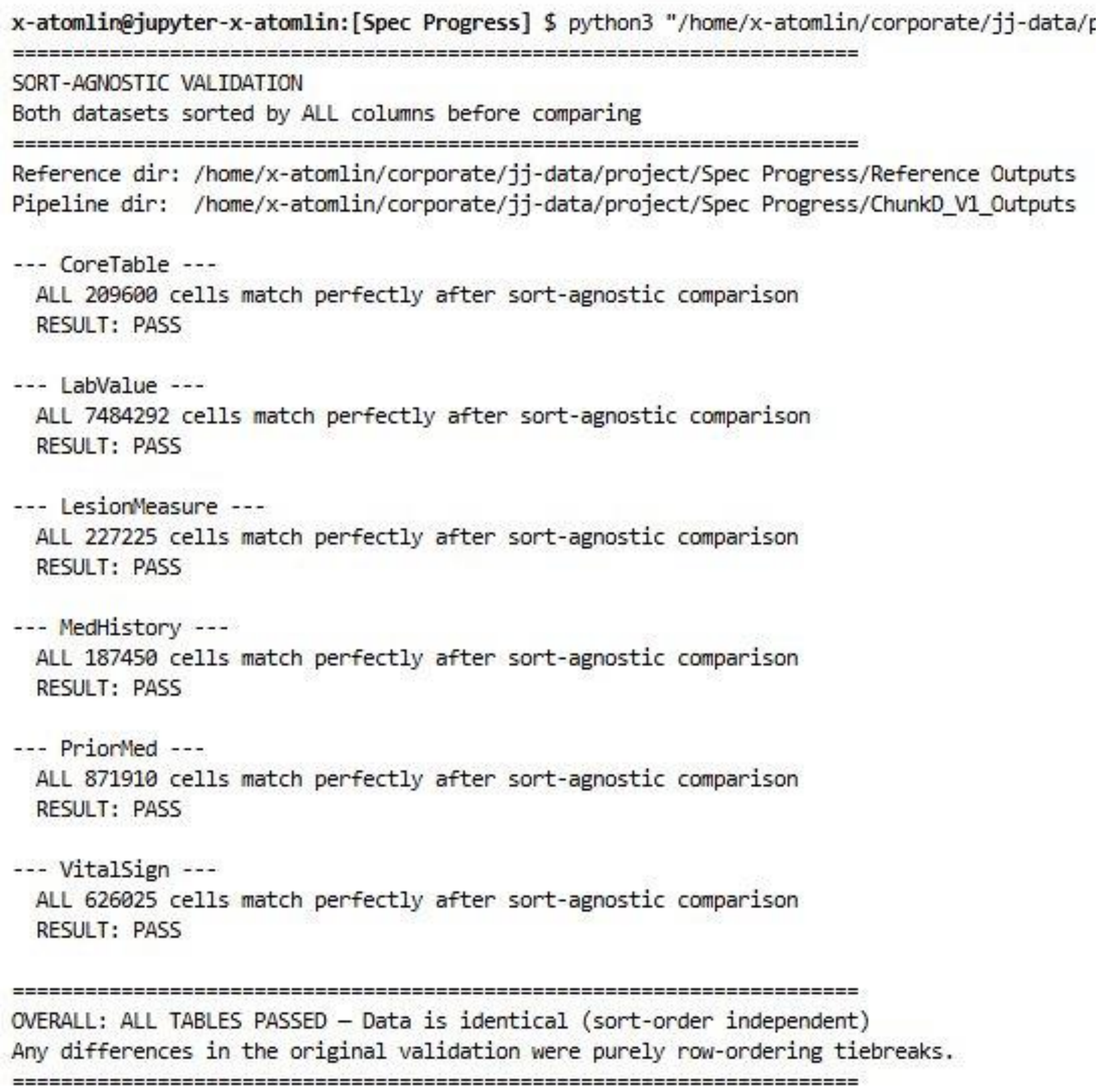
ChatGPT

RESEARCH METHODOLOGY

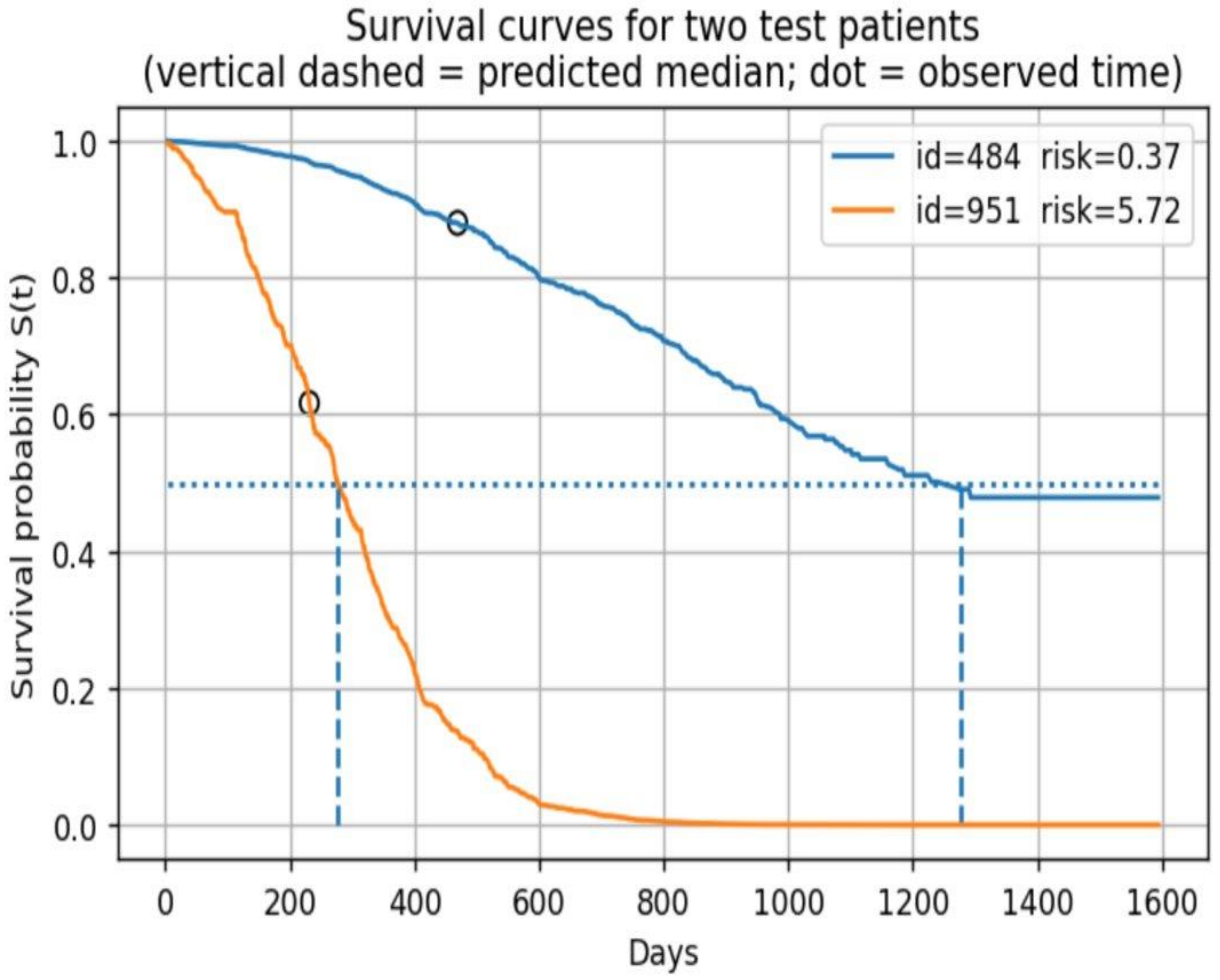


Prepare Reference Code → Write Initial Spec & Command → Generate & Compare Outputs → Diagnose with AI Assistance → Refine Spec & Command → Iterate & Document

- Collaborative iteration: Team members build on each other's improved spec/command versions, accelerating progress.
- Al-accelerated refinement: Al helps analyze discrepancies and rapidly update and re-test the spec/command, enabling fast iteration.
- Spec-driven code generation: A structured specification file drives the LLM to produce a complete, reproducible pipeline — reducing ambiguity and manual coding
- Automated validation guardrails: Generated outputs are checed against required and forbidden patterns before use, catching errors without human review of every line
- Reducing hallucinations: Splitting the pipeline into narrow-scope chunks (A-D) constrained each LLM call to a single well-defined phase



```
x-atomlin@jupyter-x-atomlin:[Spec Progress] $ python3 "/home/x-atomlin/corporate/jj-data/p
======================================================================
SORT-AGNOSTIC VALIDATION
Both datasets sorted by ALL columns before comparing
======================================================================
Reference dir: /home/x-atomlin/corporate/jj-data/project/Spec Progress/Reference Outputs
Pipeline dir:  /home/x-atomlin/corporate/jj-data/project/Spec Progress/ChunkD_V1_Outputs

--- CoreTable ---
  ALL 209600 cells match perfectly after sort-agnostic comparison
  RESULT: PASS

--- LabValue ---
  ALL 7484292 cells match perfectly after sort-agnostic comparison
  RESULT: PASS

--- LesionMeasure ---
  ALL 227225 cells match perfectly after sort-agnostic comparison
  RESULT: PASS

--- MedHistory ---
  ALL 187450 cells match perfectly after sort-agnostic comparison
  RESULT: PASS

--- PriorMed ---
  ALL 871910 cells match perfectly after sort-agnostic comparison
  RESULT: PASS

--- VitalSign ---
  ALL 626025 cells match perfectly after sort-agnostic comparison
  RESULT: PASS

======================================================================
OVERALL: ALL TABLES PASSED — Data is identical (sort-order independent)
Any differences in the original validation were purely row-ordering tiebreaks.
======================================================================
```



Survival curves for two test patients (vertical dashed = predicted median; dot = observed time)

TECHNOLOGIES AND LANGUAGES USED

LLM Models
- GPT 5.4
- Claude Opus 4
- Gemini 2.5 Pro

Data & Platforms
- Project Data Sphere (PDS) clinical trial datasets
- De-identified oncology patient data

Modeling & Statistics
- Cox's Proportional Hazards Models
- Regularized Cox (LASSO) for feature selection

Tech Stack
- Python
- Lifelines (survival analysis)
- Pandas / NumPy
- Matplotlib / Seaborn

- Differences between GenAI models: Claude Opus 4 followed the specs most faithfully and gave the best output that matched the reference code.

CONCLUSIONS

- Successfully demonstrated that a structured specification paired with precise LLM prompting can generate a complete, validated data-cleaning pipeline with no manual coding
- Automated guardrails and retry logic reduced human review time while maintaining output correctness
- Our final pipeline produced CSV outputs that matched the reference with 0 mismatches across 6 tables — over 10 million cells verified identical

FUTURE GOALS

Long-Term Vision
- A framework where the spec is a portable, auditable, and reusable product across many oncology datasets.
- Adaptable and customizable to different endpoints
- Explore using smaller, faster models like Sonnet with the same guardrails to reduce

ACKNOWLEDGEMENTS

We thank Fei Chen, Johnson & Johnson, the Data Mine staff, and Logann Tuttle for their support throughout this project!

We thank our Fall Members and TAs, Toshi Thatte, Yaduvir Singh, Akash Gaonkar, Yana Shah, Yufeng Gong, Yamana Uno, and Chaithanya Podishetti

The Data Mine Corporate Partners Symposium