Project 1: OpenClaw and Text2SQL

12412903 Xu Zhengnan

Date: 2026-04-24

1 Q 1. OpenClaw for Clinical Data Analysis

> **Remark**
> Tool selection: due to the safety reason of Claw-architecture and limitation of API, I use VSCode copilot(agent mode gemini-3.1-pro/gpt-5.3-codex) alternatively. All content passed human supervision.
>
> the prompt and agent's reply are in the appendix file.

1.1 Q 1.1: System Architecture and Working Mechanism (Autoclaw)

1.1.1 Claw introduction and system architecture

Aimed at describing my system enviroment, I chose to report my local machine information. But I suspect that due to the reset history of my system, some common instructions are not effective, like `systeminfo | findstr /B /C:"OS Name" /C:"OS Version" /C:"System Type"`.

So I use `Get-ComputerInfo | Select-Object WindowsProductName, WindowsVersion, OsArchitecture, TotalPhysicalMemory` to get the information.

Machine Information:
- Windows-10-Pro, 2009, 64-bit operating system.
- 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, 4 cores, 8 Logical Processors.

Claw-type product I have experienced (timeline):
1. nanoclaw: A minimalistic agent framework base on claudecode, which I am exploring now. Best for geeks.
2. QClaw: Tencent AI Lab's open-source agent framework, which I have explored and tested before. Not a good choice.
3. Autoclaw: Z.ai's ready to use agent system, which I used for this project. Best for beginners.
4. Clawdbot(OpenClaw's early version)

The Autoclaw (or Claw-type) system functions as an advanced autonomous agent framework designed to interact intelligently with the development environment. Most of its designs come form the famous article ReAct I think. Its core architecture consists of the following four components:

1. **Large Language Model Core (Brain)**: The central reasoning engine responsible for understanding natural language requests, formulating multi-step plans, and generating code or text.
2. **Tool Calling Interface (Hands)**: An integrated set of APIs that allows the system to read/write files, execute terminal commands, run code directly in Jupyter notebooks, and search the codebase or the web.
3. **Context and Memory Management**: Maintains conversation context, workspace state, and persistent memory (spanning session, user, and repository levels) to ensure coherence during complex and long-running tasks.

4. **Execution Loop**: Operates on an iterative "Thought -> Action -> Observation" loop. It continuously analyzes the current state, decides the next tool to call, and self-corrects based on execution feedback.

Generally, it compares strongly with the manus-type product, however, it's not the same tier product!

1.1.2 Three Skills Introduction

> **Remark**
> Here I mannually check the correctness of the skills, and **the title are clickable to open the resources!** agent behave really bad in the task. mainly because:
> 1. vscode copilot agent is not designed for skills loading, it only supports a few pre-set skills in vscode.
> 2. vscode's system prompt is quite limited, it doesn't enhance the sensitivity of skills and internet tools.
> 3. the prompt is quite obscure, it needs explicit instructions and resources. Agnet may consider it as a general human-learnable skill rather than a skill document, which means the agent mission degenerates into a general chatting session.
> 4. it doesn't suppress the tendency of ai to be lazy. ai tends to give answers from memory, which is not good for this task that needs specific and up-to-date information.

After searching and exploring various skills in the skill marketplace, we noticed a high volume of skills with duplicate naming conventions. Here, we selected the skills with the highest usage:

1. **csv-data-analysis**: This skill should be used when users need to analyze CSV or Excel files, understand data patterns, generate statistical summaries, or create data visualizations.
2. **data-scraper-agent**: Builds a fully automated AI-driven data collection agent suitable for any public source—job boards, price information, news, GitHub, sports events, etc. It scrapes on a schedule, enriches data using free LLMs (Gemini Flash), stores results in Notion/Sheets/Supabase, and learns from user feedback. It runs completely free on GitHub Actions and is ideal for users who want to automatically monitor, collect, or track public data.
3. **visualization-expert**: Chart selection and data visualization guidance for effective data communication. Use when creating visualizations, choosing chart types, designing dashboards, or when the user mentions data visualization, charts, graphs, or needs help presenting data visually.

Observation: High numbers of "Stars" and "Forks" primarily originate from official skills published by the platform or high-quality integration libraries (e.g., everything-claude-code). Foundational skills like data analysis inherently occupy a large portion of the platform. The challenge lies in finding more specialized, high-quality skills. It is concluded that using the platform's general-purpose skills is sufficient; using others without specific needs simply wastes context space.

1.2 Q1.2: Spark-Based Descriptive Analysis of Diabetes Risk

1. **Highly Imbalanced Class Distribution**:
   - `Diabetes_012 = 0` (No Diabetes) accounts for **84.24%**.
   - `Diabetes_012 = 1` (Prediabetes) accounts for **1.83%**.
   - `Diabetes_012 = 2` (Diabetes) accounts for **13.93%**.
2. **Consistency Between Spark SQL and DataFrame APIs**:

- Comparing the outputs, the differences between the indicators are exactly 0.0, indicating that both APIs yield identical conclusions for this statistical task.

3. **Relationship Between Health Indicators and Diabetes Risk (by Category Mean)**:
   - `HighBP` : Increases from $0.371 to 0.629 to 0.753$, showing a clear increase with higher diabetes severity.
   - `HighChol` : Increases from $0.379 to 0.621 to 0.670$, similar to blood pressure.
   - `BMI` : Increases from $27.74 to 30.72 to 31.94$. The diabetic group has a higher average BMI.
   - `PhysActivity` : Decreases from $0.779 to 0.678 to 0.631$. Higher diabetes risk corresponds to a lower proportion of physical activity.
   - `Age` : Increases from $7.79 to 9.08 to 9.38$. The diabetic group generally falls into higher age brackets.
4. **Overall Conclusion (Most Relevant Factors)**:
   - Among these five indicators, `HighBP` , `HighChol` , and `BMI` show a strong positive correlation trend with diabetes risk.
   - `PhysActivity` is negatively correlated with the risk.
   - `Age` also displays a clear positive correlation.
   - *Note:* The descriptive analysis findings reflect correlational trends rather than causal relationships.

1.3 Q1.3: Three Observations with Codes, Tables, and Graphs

Below are data-driven observations from this dataset. Each observation translates the findings into analytical outcomes. Codes is the notebook files.

1. Conclusion 1: HighBP and HighChol both increase strongly from class 0 to class 2.
2. Conclusion 2: Individuals in higher diabetes classes are older on average and have higher BMI.
3. Conclusion 3: Physical activity rate drops by 0.149 from class 0 to class 2.

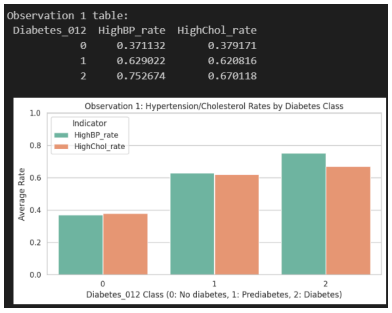

Figure 2: Fig. 1a: HighBP and HighChol increase with diabetes class.

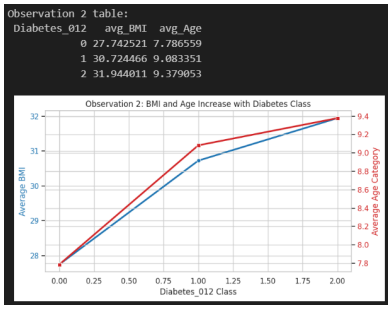

Figure 3: Fig. 1b: BMI and Age increase with diabetes class.

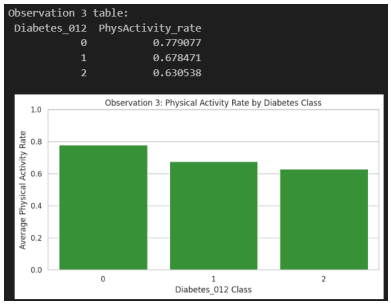

Figure 4: Fig. 1c: Physical activity decreases with diabetes class.

Figure 1: Three key observations from Spark-based descriptive analysis.

As shown in Figure 2, Figure 3, and Figure 4, the panel in Figure 1 supports a consistent risk pattern across cardiometabolic indicators, BMI/Age, and physical activity.

Observation Note: Higher diabetes classes are invariably associated with a significantly higher prevalence of high blood pressure and high cholesterol.

1.4 Q1.4: Failure cases analysis

> **Remark**
> These part is mentioned before, now presented in the format required and add some insights in the analysis.

- case: desire agent(model:gemini-3.1-pro) generate skills introduction directly without human supervision.
- object: desire high quality answers without faciles.
- expected result: expect agent tool generate high quality results, including explicit skills description, verifiable recourses link, and the reasonable choices based on the sorts of stars or forks numbers.
- unsatisfactory result: general and vague description of the skills, no resources link, and no reasonable choices. Totally rubbish and not helpful for the task.
- analysis:
  1. model has the tendency of being lazy, it prefers to give answers from memory rather than actively searching for up-to-date information. This is a common issue with language models, especially when the prompt is not specific enough to guide them towards using tools or accessing external resources.
  2. the prompt provided to the model may not have been clear or specific enough to encourage it to utilize the skills marketplace effectively. If the prompt is too vague, the model might default to generating general information based on its training data rather than performing the necessary steps to access and analyze the skills marketplace.
  3. the model may have misunderstood the task or the concept of "skills" in this context, leading to a failure in generating the expected results. This could be due to ambiguity in the prompt or a lack of understanding of how to use the skills marketplace effectively.
  4. the model's architecture and system prompt may not have been designed to handle this specific type of task, which could have contributed to its poor performance. If the model is not optimized for tasks that require accessing external resources or using specific tools, it may struggle to generate high-quality results in this context.
- valuable aspect: when handling tasks that require specific and tender attentions, the agent tool's architecture must be suitable to the task's characteristics. For example, in this case, the copilot agent is not designed for skills loading and has limited support for online searching, which makes it less effective for tasks that require accessing and analyzing the skills marketplace or creating skill-docs in hand if you need to use SKILLs. On the other hand, a tool like OpenClaw, which is specifically designed to support skills loading and has better online search capabilities, would likely perform much better in this context. This highlights the importance of choosing the right tool for the task at hand and ensuring that the tool's architecture is well-suited to the specific requirements of the task.

2 Q 2. Survival Analysis

First I use `/opt/venv/bin/python /tmp/dbx_py_to_ipynb.py && ls -1 /root/dev/project1/sur/*.ipynb` to convert the python script to ipynb file, and then open it in jupyter notebook to run the code and generate the results.

2.1 Q 2.1/2.2: Spark-Based Survival Analysis and Interpretation.

2.1.1 Objective and Workflow

This case reproduces the survival-analysis pipeline for telecom churn using a hybrid Spark + lifelines workflow. The objective is to estimate customer survival, identify churn risk drivers, and convert survival probability into business value (CLV).

The practical workflow is:

1. **Data ingestion (Spark Bronze layer):** load `Telco-Customer-Churn.csv` with an explicit schema.
2. **Data filtering (Spark Silver layer):** keep only `Month-to-month` customers with active internet service, and convert `churnString` into numeric event indicator `churn`.
3. **Kaplan-Meier (KM):** estimate non-parametric survival curve and median survival time.
4. **Cox Proportional Hazards:** estimate covariate-level relative churn risk.
5. **Weibull AFT:** model time-to-event parametrically for forward projection.
6. **CLV conversion:** transform predicted survival probabilities into expected cumulative revenue.

2.1.2 Data Preparation Results

- Input dataset: `Telco-Customer-Churn.csv`.
- Target analysis segment after Spark filtering:
  - contract == `Month-to-month`
  - internetService != `No`
- **Silver layer valid rows: 3351**
- Event definition:
  - `churn = 1.0` if original label is `Yes`
  - `churn = 0.0` if original label is `No`

> **Remark**
> Execution note: Spark reported a CSV-header naming mismatch warning (`Churn` vs schema field `churnString`, plus case differences in several columns), but the pipeline completed successfully and produced valid downstream model outputs.

2.1.3 Kaplan-Meier Survival Results

KM was fitted on:

- duration variable: `tenure` (months)
- event variable: `churn`

Recorded result:

- **Median Survival Time = 34.0 months**

Interpretation: At around 34 months, the estimated survival probability falls to 0.5, meaning about half of the target month-to-month internet customers are expected to have churned by that horizon.

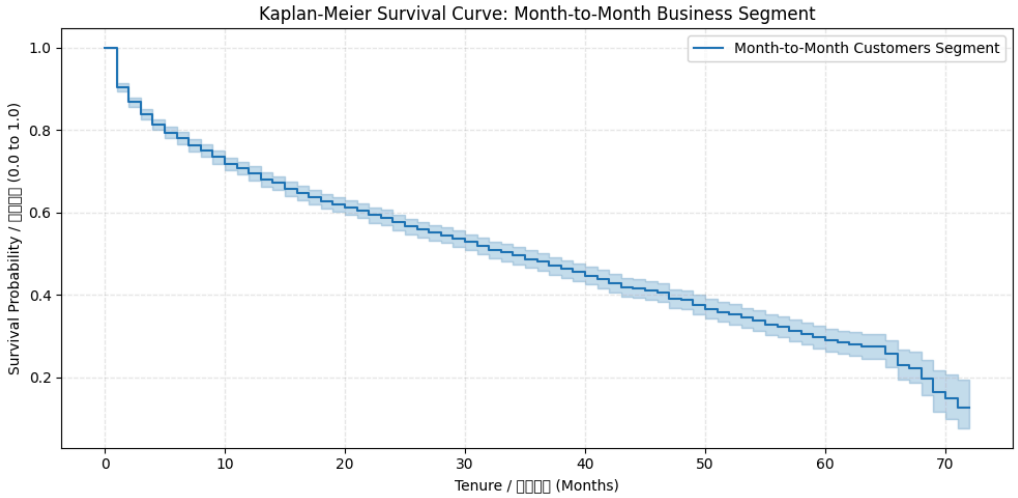

Figure 5: Kaplan-Meier survival curve for the month-to-month internet segment.

The empirical survival decay pattern is shown in Figure 5.

2.1.4 Cox PH and Weibull AFT Results

For multivariate modeling, categorical variables were one-hot encoded (`pd.get_dummies(..., drop_first=True)`), and penalization (`penalizer = 0.01`) was used to stabilize estimates.

Cox PH interpretation summary (from coefficient plot):

- Positive-side factors (higher churn hazard tendency):
  - `internetService_Fiber optic`
  - `phoneService_Yes`
  - `paymentMethod_Electronic check`
  - part of entertainment/billing related dummies
- Negative-side factors (lower churn hazard tendency):
  - `onlineSecurity_Yes`
  - `techSupport_Yes`
  - `onlineBackup_Yes`
  - `partner_Yes` (milder effect)

Weibull AFT interpretation summary (from parameter-effect plot):

- Protective service variables (`onlineSecurity_Yes` , `techSupport_Yes`) are associated with longer expected survival time.
- Risk-prone factors (notably `internetService_Fiber optic` , `paymentMethod_Electronic check`) shift the model toward faster failure/churn timing.

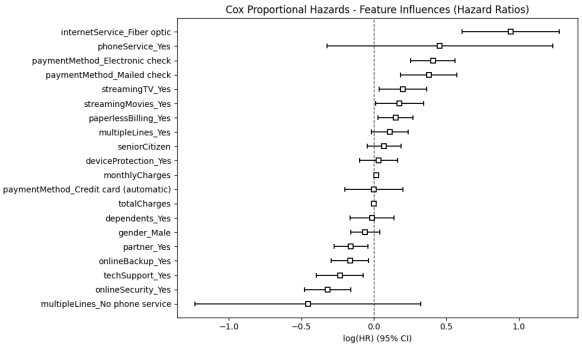

Figure 7: Cox PH feature influences (hazard ratios).

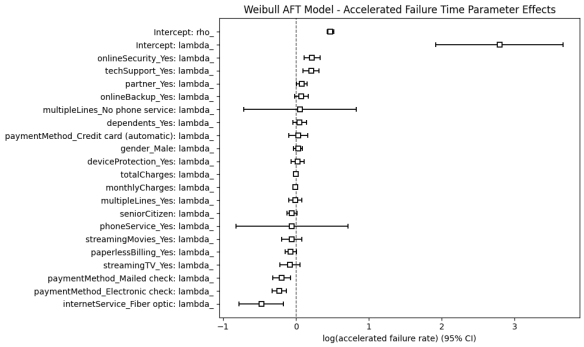

Figure 8: Weibull AFT parameter effects.

Figure 6: Risk-factor diagnostics from Cox PH and Weibull AFT models.

The multivariate risk decomposition is shown in Figure 7 and Figure 8 (combined panel: Figure 6).

2.1.5 CLV (Customer Lifetime Value) Results

Using AFT-predicted survival for future months $t = 1..36$, the report converts survival into expected revenue:

- **Average monthly revenue (ARPU): $73.59**
- **Expected cumulative CLV at 12 months: $653.25**
- **Expected cumulative CLV at 24 months: $1027.92**
- **Expected cumulative CLV at 36 months: $1297.28**

Interpretation: The cumulative CLV curve increases over time but with diminishing slope, reflecting declining survival probability as horizon extends.

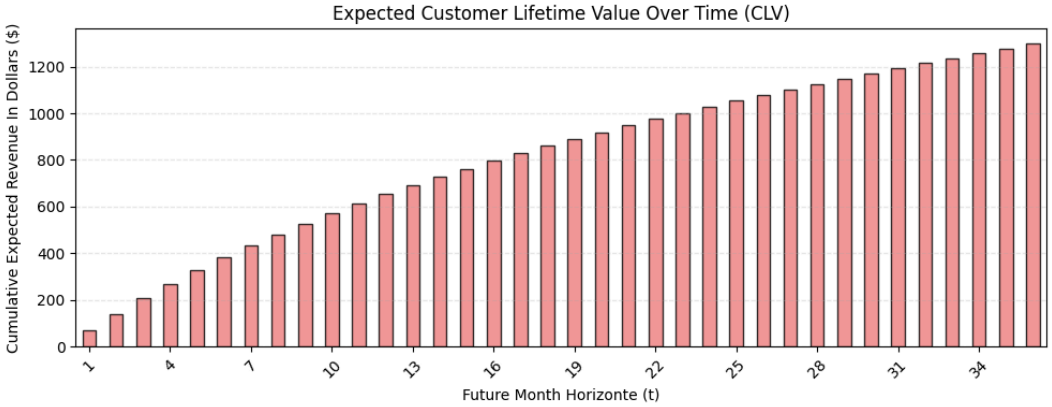

Figure 9: Expected cumulative customer lifetime value over 36 months.

The monetized retention trajectory used for business budgeting is presented in Figure 9.

2.1.6 Business Interpretation and Actionable Insights

1. **Retention priority targeting:** Month-to-month internet users form a high-volatility segment and should be managed as a dedicated risk cohort.
2. **Most actionable risk drivers:** electronic-check payment channel and fiber-optic segment exposure show stronger association with churn acceleration in this sample.
3. **Most actionable protective drivers:** enabling security/support-related services is associated with longer customer lifetime.

4. **Budget guardrail from CLV:** interventions for this segment should be evaluated against the projected CLV ceiling (e.g., 12/24/36 month benchmarks above) to avoid overspending on retention CAC.

2.2 Q 2.3: MySQL Tests and Failure Cases Analysis

> **Remark**
> When investigating the MySQL-based LLM-SQL testing, I aligned the evaluation framing with recent Text-to-SQL pipeline research [1].

2.2.1 MySQL Testing Summary

The SQL testing setup follows the Text-to-SQL evaluation idea in [1]. I used one challenge set (`telco_sql_challenges.sql`) and one answer/fix file (`answers.sql`) for reproducible verification.

Module	What was executed	Key output / decision
Environment	MySQL 8.0 local DB + `telco_churn` + CSV import + indexes	Ready for LLM SQL testing
Question set	Q1-Q7 in `telco_sql_challenges.sql`	Q6-Q7 are stress tests
Answer set	`answers.sql` + fixed SQL block	Used as candidate answers + patches
Core benchmarks	Churn rate, contract ranking, OR vs DSL, retention curve	Numerically stable and reproducible
Q6 result	IPW-style uplift by contract	`Month-to-month` +8.81 pct; proxy only
Q7 result	Hazard-style month table with CI	Computable from snapshot, not fully identifiable

2.2.2 Failure Cases Analysis

- **Case 1: Import security block (`LOAD DATA LOCAL INFILE`)**: Trigger: `ERROR 3948` (server/client local infile disabled).
  Impact: CSV import failed in first run.
  Fix: enable `SET GLOBAL local_infile = 1` and run client with `--local-infile=1` .
- **Case 2: Metric-definition mismatch in validation text**: Trigger: `Q2 baseline Odds` mixed **odds** and **odds ratio** interpretation.
  Impact: validation statement could mislead result judgement.
  Fix: either keep odds and update expected text (0.23), or compute explicit self-ratio (=1.00).
- **Case 3: Executable SQL errors in Q6/Q7 validation CTEs**: Trigger: `Unknown column 'events'` and latent missing-column reference for `payment_method` .
  Impact: full validation block could not execute end-to-end.
  Fix: project `events` in hazard subquery and carry/aggregate `payment_method` at the correct CTE level.
- **Case 4: Causal over-interpretation risk**: Trigger: SQL-only uplift/hazard outputs from one-row snapshot table.
  Impact: values are easy to overclaim as causal truth.
  Fix: mark outputs as **proxy estimates** and require extra assumptions/data for causal claims.

2.2.3 Assumptions: When LLMs May Fail in SQL Data Analysis

The following assumptions describe typical conditions under which LLM-generated SQL is likely to fail:

1. **Column-lineage fragility in long CTE chains**:
   - Assumption: if a query spans many nested CTEs, the model may reference columns that are not projected in intermediate steps.
   - Evidence in this task: `events` and `payment_method` reference errors in the Q6/Q7 validation block.
   - Paper viewpoint [1]: pipeline SQL generation relies on staged correction/merge correction exactly because one-pass SQL often contains logic/syntax defects; intermediate-stage consistency is critical.
2. **Metric-definition drift**:
   - Assumption: the model can mix related but different metrics (e.g., odds vs odds ratio, rate vs probability) while keeping SQL syntactically valid.
   - Evidence in this task: DSL baseline check text mixed raw odds with self-ratio expectation.
   - Paper viewpoint [1]: the paper evaluates with execution accuracy (EX) rather than string matching, implying semantic correctness of result is more important than superficially plausible SQL text.
3. **Identification vs computation confusion**:
   - Assumption: for causal/uplift questions, the model may produce computable SQL but overclaim causal interpretation without sufficient assumptions.
   - Evidence in this task: SQL-only IPW-style output is a proxy estimate, not an identified causal effect.
   - Paper viewpoint [1]: BASE-SQL targets Text-to-SQL execution correctness, not causal identification; therefore, correctly executable SQL does not automatically imply causal validity.
4. **Data-generating-process mismatch**:
   - Assumption: if the table is a static snapshot, the model may still attempt full time-varying hazard reconstruction as if event-history data existed.
   - Consequence: outputs can be numerically plausible but not statistically identifiable.
   - Paper viewpoint [1]: schema representation and schema-linking quality strongly affect downstream SQL quality; missing/insufficient temporal schema information should be treated as a hard limitation, not something prompting can fully recover.
5. **Weak validation habit**:
   - Assumption: the model may stop at "query runs" and omit consistency checks (sum checks, range checks, overlap checks, baseline checks).
   - Recommendation: always pair each analytical SQL with explicit validation SQL.
   - Paper viewpoint [1]: SQL revision + merge revision improve EX in ablations, supporting the need for post-generation verification loops instead of single-pass generation.

Practical takeaway: LLM-generated SQL should be treated as a draft requiring schema-level review, metric-definition audit, and validation-query confirmation before analytical conclusions are accepted. We need to use the PLAN agent first or do small modules implementation to improve the SQL code quality.

2.3 Q 2.4: Personal Website

Remark
I have built a personal website using Next.js before! name.com, namecheap, aliyun cost a an arms and a leg and it's not necessary. As a student developer, I firmly chose Github Pages!

Remark
website: https://chengou-zheng.github.io/

Here I also upgrade my website from ananke into hugo book, whose wiki style maybe more suitable for my future blogging and project report.

Here is a direct comparison between the two themes:

Figure 11: v1: ananke theme.

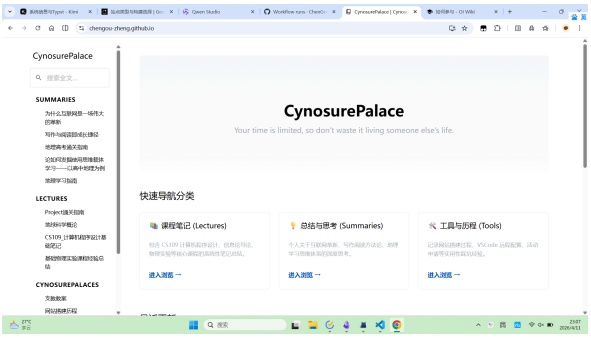

Figure 12: v2: hugo book theme.

Figure 10: personal website theme comparison.

3 Appendix: Project Materials Checklist

Category	Filename	Purpose / Description	Status
Final report	project1.typ	Main editable source of this report (Typst).	Ready
Final report PDF	project1.pdf	Compiled report PDF for submission.	Ready
Notebook (Q1)	project1.ipynb	Spark descriptive analysis and visualization for diabetes dataset.	Ready
Notebook (Q2)	survival.ipynb	End-to-end survival analysis workflow (KM, Cox, AFT, CLV).	Ready
SQL answers	answers.sql	LLM challenge answers, validation SQL, and corrected SQL snippets.	Ready
MySQL setup	telco_mysql_setup.sql	One-shot database/table creation and CSV import script.	Ready
SQL challenge set	telco_sql_challenges.sql	High-difficulty SQL questions for LLM evaluation.	Ready

Raw churn data	Telco-Customer-Churn.csv	Source dataset used in MySQL and survival-analysis tasks.	Ready
Q1 data package	Q1_data/	Diabetes indicator dataset and related input files.	Ready
Figure assets	assets/	All exported figures used in the report (Q1 and Q2).	Ready
Presentation slides	project1-presentation.pdf	Presentation deck for project defense/reporting.	Ready
LLM interaction history	llm_interactions.txt	Raw prompt and response logs from LLM interactions during the project.	Ready

4 References

Bibliography

[1] L. Sheng, S.-S. Xu, and W. Xie, "BASE-SQL: A powerful open source Text-To-SQL baseline approach," *arXiv preprint arXiv:2502.10739*, 2025, [Online]. Available: https://arxiv.org/abs/2502.10739