GE Capstone Mid Report – LLMs for Time Series

Zhiqi Ma, Bingqian Lu, Yangruonan Lin, Ruobing Zhang
Columbia University, DSI

Contents

1 Introduction

1.1 Project Scope and Goal

This project is guided by GE (General Electric), a high-tech industrial company engaged in aerospace, renewable energy, and power business. The core objective is to leverage Large Language Models (LLMs) for multi-modal data analysis and notification generation. Specifically, it aims to bypass traditional coding methods and use natural-language specifications to LLMs to analyze multi-modal datasets, which contain time series numerical and text data (among other types) from corporate operations and asset monitoring databases.

The project will assess the capabilities and limitations of state-of-the-art LLMs, especially in the domain of time series tasks. It will generate notifications based on natural-language requirements when certain events are detected, such as "Notify the maintenance staff if the asset's average temperature has risen by 5 degrees over a week and no maintenance has been conducted in the past six months." Given that data privacy is one of the significant concerns for the company (e.g. in aerospace and power industry), this project also pays special attention to models which can be deployed locally to tackle data privacy issues.

Specifically, three major tasks related to time series analysis will be examined, including threshold exceedance, slope calculation, and forecasting, to evaluate the performance of LLMs using standard Machine Learning metrics. Overall, the project seeks to shift from traditional software development processes, as depicted in the V-model (involving requirement gathering, system analysis, software design, etc.) in Figure 1, to a process driven by LLMs and prompts, thereby simplifying tasks, facilitating data exploitation, and enabling fast prototyping.

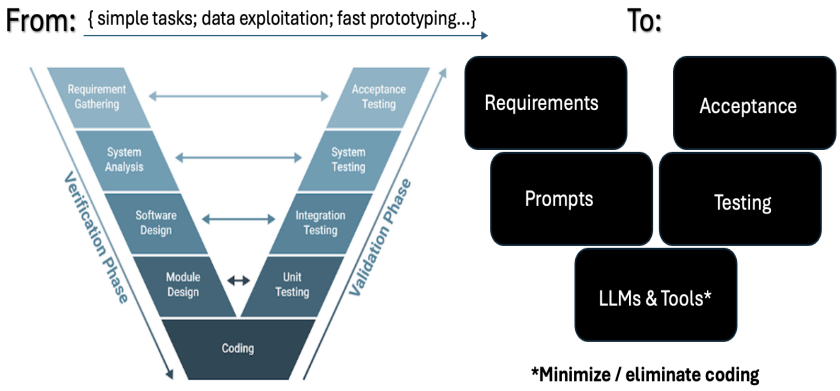

Figure 1: High Level Project Goal

1.2 Literature Review

We conducted a literature review to explore the integration of large language models (LLMs) into time series related tasks. The following six papers were mainly analyzed:

LLMTIME (used by previous team) carefully tokenizes time series as text, converts discrete distribution into continuous likelihood, enabling zero-shot learning without fine-tuning, which is computationally efficient. It outperforms traditional time series models in certain scenarios but may face challenges such as arithmetic precision and handling complex patterns.[2]

TIME-LLM also employs a reprogramming approach to align time series data with LLMs. By transforming time series into text prototypes and using Prompt-as-Prefix (PaP) technique, TIME-LLM achieves good performance in few-shot and zero-shot settings.[3]

FPT uses pre-trained language models for general time series analysis. By freezing the self-attention and feedforward layers of the pre-trained model and fine-tuning the remaining layers, the model achieves competitive performance in various time series tasks.[8]

TIME-MOE introduces the largest time series foundation model with a mixture-of-experts decoder-only transformer architecture, using the sparse activation mechanism to decrease computational cost. It also employs multi-resolution forecasting to better capture data complexities and supports inputs of arbitrary lengths for any-variate task. Trained on Time-300B dataset (also the biggest), it achieves SOTA performance in both zero-shot and full-shot time series forecasting.[6]

CHRONOS is a simple yet effective framework for pretrained probabilistic time series models. It tokenizes time series into a fixed vocabulary and trains language models on them. Trained on a large collection of datasets, it significantly outperforms traditional and task-specific models in in-domain forecasting. In zero-shot forecasting, it performs on par with the best deep-learning models, showing great potential in simplifying forecasting pipelines.[1]

TIMERAG integrates Retrieval-Augmented Generation (RAG) into time series forecasting. It segments data, clusters similar segments, and constructs a time series knowledge base, then uses dynamic time warping for sequence retrieval, finally using the method from TIME-LLM for forecasting tasks. TIMERAG outperforms TIME-LLM and several SOTA benchmarks of the time.[7]

These studies offer valuable insights of the integration of LLMs into time series tasks. Each model has distinct characteristics and advantages, and they contribute to the development of more accurate and efficient time series forecasting methods. Our project is inspired by these works and aims to further explore the potential of LLMs in time series analysis.

2 Data

2.1 Predictive Maintenance

The Microsoft Azure Predictive Maintenance (PdM) dataset utilized in this project serves as an exemplary data source for predictive maintenance model building, i.e., for analyzing machine-related conditions and behaviors.

Telemetry Time Series Data: Sensor-collected time series data (*PdM_telemetry.csv*) provides hourly average values of voltage, rotation, pressure, and vibration throughout 2015 for 100 machines. These data offer real-time insights into machine performance during operation.

Error and Failure history: Errors (*PdM_errors.csv*) are encountered by the machines while in operating condition, while failures (*PdM_failures.csv*) represent the replacement of a component due to failure. Failures data is a subset of the Maintenance data. This information is vital for understanding the frequency and nature of machine malfunctions.

Maintenance history: *PdM_maint.csv* includes the repair history of machines. It details Machine ID and components for replacements both during scheduled visits (Proactive Maintenance) and unscheduled maintenance (Reactive maintenance) caused by component breakdowns (failures). The maintenance data, spanning 2014 - 2015, is rounded to the nearest hour to match the telemetry data's collection frequency.

PdM dataset is suitable for conducting in-depth time-series analysis related to predictive maintenance. The telemetry data is used for all three time-series tasks. Specific subsets are extracted based on different machine groups and variables. The remaining 3 datasets are mainly used for the threshold exceedance task.

For the slope calculation task, the original telemetry dataset had little variation, resulting in slopes close to zero. To address this, trends and noise were manually added. A positive slope and a negative slope were added to subsets of telemetry respectively, as shown in Figure 2 and Figure 3. The negative slope was included only for better evaluation with no real-world meaning.

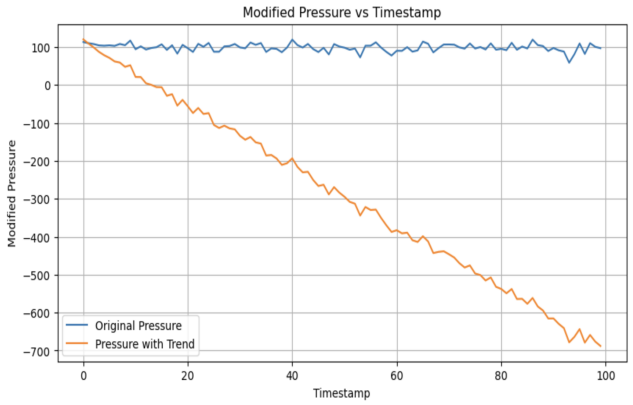

Figure 2: Modification with Negative Slope

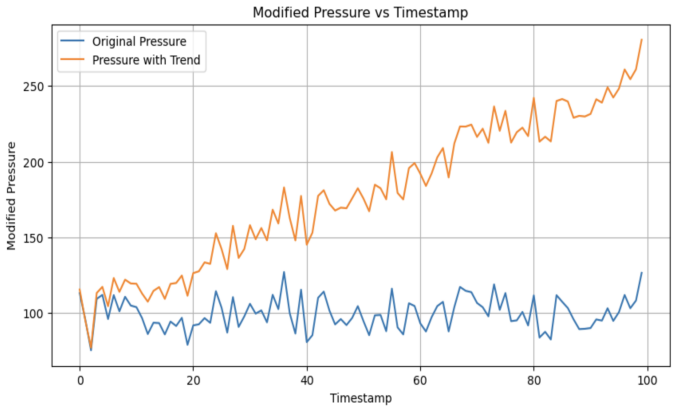

Figure 3: Modification with Positive Slope

2.2 Text data

The text data used in the project is synthetic and generated by the same LLM model powering the agent. It consists of machine maintenance records, including descriptions of maintenance tasks, machine status, and machine IDs. This data is used to evaluate the model's ability to analyze text data, specifically its capacity to accurately filter out records requiring maintenance. In this project, we

generated two sets with different description lengths, one with shorter description lengths, and the other with longer description lengths, for the completeness of the evaluation.

2.3 Other Datasets

Though PdM dataset is most relevant to the goals of this project, for the concerns regarding the dataset's limited variability and the LLMs models' generalizability in a broader context, other time series datasets across different domains and temporal resolutions are chosen for further testing. This includes datasets from Monash Time Series Forecasting Repository, DARTS Library, and UCI Machine Learning Repository, which are well-established resources for time-series data, and possibly synthesized datasets can also be used.

For a complete list of the datasets, please refer to Appendix A (Dataset List) for details.

3 Time Series Tasks

Three key time-series tasks were defined and investigated in this project to evaluate relevant models. These tasks were not limited to the Predictive Maintenance (PdM) dataset, with more focus on the forecasting task for other datasets.

3.1 Threshold Exceedance

This task is crucial for identifying critical events in time-series data. It involves detecting when specific values cross predefined thresholds. In industrial applications like power plants, monitoring steam turbine pressure thresholds can prevent catastrophic failures. In financial markets, if the stock price drops below a certain threshold set by risk-management strategies, it may trigger actions such as selling the stock to limit losses.

Choice of threshold requires careful consideration of the data's nature, and thresholds are typically set by experts based on professional knowledge and past experience. Yet due to limited conditions in this project, we had to set values manually, such as using 95% percentiles or mean standard deviations methods.

During evaluation, the number of values exceeding the threshold in the dataset is counted, and the LLMs' performance is evaluated based on their **Accuracy** (percentage of true exceedance detected by the model) in detecting these exceedances.

3.2 Slope Calculation

This task helps analyze trends and patterns in time - series data by determining the rate of change over a specific time interval. For example, in environmental monitoring, it can reveal climate change trends, and in business analytics, it can show sales growth or decline. The slope calculation task not only helps in understanding past and current trends but also serves as a basis for making informed predictions about future data behavior. It also aids in prediction and can be combined with other tasks. For example, data points with a rate of change more extreme than the average slope during an interval may be alerted as an anomaly that exceeds the threshold; while forecasting can also be used to predict future slope based on current values.

For model evaluation, the ground truth slope is calculated using Linear Regression, and then then LLMs' calculated slopes are compared using **MSE** (Mean Squared Error) and **MAE** (Mean Absolute Error) metrics.

3.3 Forecasting

This is a fundamental task in time-series analysis, aiming to predict future values based on historical data., and in fact most of current research in time-series domain devotes to this topic. In weather forecasting, meteorologists use historical temperature, humid-

ity, and wind data to predict future weather conditions. In the field of e-commerce, forecasting future sales is crucial for inventory management. Different forecasting methods, such as auto-regressive integrated moving average (ARIMA), neural networks, or more advanced deep-learning-based models, can be employed depending on the complexity of the data and the required accuracy, while this project mainly focus on LLMs models.

During evaluation, forecasted values from LLMs are compared with the actual time-series data using metrics like **MAE**, **RMSE** (Root Mean Square Error), and **MAPE** (Mean Absolute Percentage Error).

4 Modeling Approach

4.1 Modeling Overview

In line with the project's goal, several LLM-based models were selected.

On one hand, we aimed to explore the cutting-edge time-series analysis research. After extensive literature reviews, we found that most current research focus on time-series forecasting exclusively. Thus, to perform tasks other than forecasting, we need to first design auxiliary tasks based on forecasting that can indirectly reflect the model's performance on those other tasks. Plus, most current approaches for time-series LLMs lack support for prompt-based interaction or cannot accept textual data as input. Nevertheless, since it's very likely these advanced research and models will be extended to include additional features just mentioned in the near future, we still chose one such model.

In this direction, the most suitable candidate model is Time-MoE. At least from the model's paper, it outperforms all other time series foundation models as well as advanced deep learning and statistical approaches for time series forecasting, with another highlight is its MoE architecture, which is extremely efficient and quite novel to time-series domain. However, unfortunately, since the model is so new (released on Jan-2025), its incomplete features and APIs made it unusable. Thus, Amazon's Chronos model was considered a good alternative, especially with its newly released Bolt version.

On the other hand, we strived to bypass coding throughout the project – we sought to convey task requirements to models via prompt-based instructions and enable the LLMs to execute tasks automatically.

To this end, we selected general-purpose LLMs like GPT o3 -mini and DeepSeek R1. The LangChain Pandas Dataframe Agent was chosen for its ability to break down complex tasks into subtasks for LLMs, simplifying time-series analysis workflows. It had proven effective in last semester for this project, and we aimed to expand its use by integrating more diverse text data.

In combination, such a dual-objective approach was designed to both keep pace with the latest trends in time-series analysis research and to simplify the operational process leveraging LLMs' capabilities.

4.2 Chronos

Chronos, developed by Amazon in 2024, is a pre - trained foundation model family for time series forecasting, based on LLM structure (T5 architecture).[5] It differs from the original T5 mainly in vocabulary size, having fewer parameters.

In the original Chronos model, time series are tokenized by scaling and quantization and a language model is trained on them using cross-entropy loss. It uses TSMixup and KernelSynth to augment data quality. Chronos models have been trained on a 84B publicly

available time series data as well as synthetic data. Once trained, the model predicts time series using zero-shot probabilistic forecasts by sampling multiple future trajectories given the historical context. The newly released (in the end of Dec-2024) Chronos-Bolt models use patching (segment a time series into several patches) instead of tokenization, which increases the inference speed for about 250 times faster with higher accuracy. The Chronos-Bolt family offers four models of different sizes for various forecasting complexities, from 9M (Chronos-Bolt-tiny) to 205M (Chronos-Bolt-base).

AutoGluon-TimeSeries (AG-TS) provides an easy way to use Chronos nd it can be installed and imported as autogluon library on Google Colab. Roughly speaking, we can:
- Use Chronos in zero-shot mode to make forecasts without any data-specific training;
- Fine-tune Chronos models on custom data to improve the accuracy;
- Handle covariates & static features by combining Chronos with a tabular regression model.
In this project, only zero-shot forecasting and fine-tuned forecasting will be examined, as covariate analysis may add too much complexities and is out of the scope of other pure-LLM models.

Since Chronos only supports forecasting, for threshold exceedance and slope calculation, we first forecast and then use the predicted data for evaluation:
- For threshold exceedance: we count the exceedance on the predicted data points we just collected and compared with the ground truth for evaluation;
- For slope calculation: we calculate the slope on the predicted data points and again compare with the ground truth.

Rough Performance:

So far we tested Chronos on two subsets of the PdM telemetry dataset for zero-shot and fine-tuned forecasting. Dataset 1 contains 1000 data points (using the first 800 to predict the next 200), and Dataset 2 has 5000 data points (using the first 4872 to predict the next 128), both for univariate time series forecasting of the 'Pressure' variable. Dataset 2 was chosen because: as listed in Chronos' paper, most of history lengths used for test are among 3000 to 30000 (much longer than 1000 in Dataset 1), while prediction lengths are mostly 24, 48, and even shorter. So we used much longer data (5000 data points in total) to see if longer history will lead to better results.

1. Zero-shot forecasting performance:

In **Dataset 1**, both Chronos Bolt small and Chronos Bolt base models outperform Prophet, albeit marginally (0.02, 0.02, 0.0003 improvement for **MAE**, **RMSE**, and **MAPE** respectively), with the Bolt base model performing better. For the threshold exceedance task based on zero-shot predictions, all three models achieve an **accuracy** of 0.985. When it comes to slope calculation on a modified dataset, the Bolt small model doesn't perform well, but the Bolt base model is more promising compared with Prophet (0.15, 0.42, 0.0001 drop for **MAE**, **RMSE**, and **MAPE** respectively), with its forecasting line almost matching the original, resulting in similar slope and intercept values to the ground truth.

In **Dataset 2**, Chronos models access more historical data, leading to more conservative forecasting (remind that Chronos gives probabilistic forecasts from a set of possible future trajectories), which decreases the model performance a little bit but gives more cautious forecasting results, though the performance difference is minor: between Chronos Bolt small and Prophet, we have a 0.07 drop in **MAE** (7.375644 vs. 7.297019), 0.08 drop in **RMSE** (9.569981 vs. 9.476195), and 0.001 drop in **MAPE** (0.075602 vs. 0.074331); between Chronos Bolt base and Prophet, we have a 0.06, 0.07, 0.001 drop in **MAE**, **RMSE**, and **MAPE**, respectively. Chronos models perform better than Prophet in the

threshold exceedance task (0.01 improvement on **accuracy**). However, for Slope Calculation task, since we have 5000 data points, the large magnitude of data points to be predicted (around 9000 to 10500) puts pressure on forecasting for both Chronos and Prophet, though the estimated slopes are still quite accurate. And Chronos models perform slightly worse than Prophet (2.2, 2.7, 0.0012 drop in **MAE**, **RMSE**, and **MAPE**, respectively).

2. Fine-tuning forecasting performance:

Note: Chronos provides default fine-tuning parameters, but usually do not achieve better results than zero-shot, especially if the dataset is complicated (which is the case here for PdM telemetry data), so we need manual hyper-parameters setting.

In **Dataset 1**, manual hyper-parameter tuning for large models (Bolt base & Bolt small) does have small improvement, but expanding the hyper-parameter space (more hyper-parameters for models to choose) may reduce performance, likely due to overfitting and insufficient history. Smaller models (Bolt mini & Bolt tiny), however, see a significant performance boost from fine-tuning and outperform Prophet (0.15, 0.12, 0.002 improvement for **MAE**, **RMSE**, and **MAPE** respectively). Changing the context length affects smaller models' performance, with shorter lengths yielding better predictions, perhaps because longer contexts make fine-tuning more difficult and lead to overfitting.

Situations are different for Slope Calculation: For positive slope, fine-tuned Bolt mini performs better than larger models, but Bolt tiny performs poorly. The **MAPE** performance for tuned Bolt mini is about 0.48% while Prophet gives 0.40%, already quite accurate. But for negative slope, larger models are more accurate again, and fine-tuning performs worse than zero-shot – Bolt small gives 0.7% while Prophet gives 0.4% for **MAPE**. Things get complicated now due to the negative slope and relatively short history data for models to learn.

In **Dataset 2**, all fine-tuned Chronos models show improved performance, outperforming Prophet in forecasting. With longer history, overfitting issue seems to almost disappear (may still exist for Bolt base) – now bigger models achieve performance no worse than smaller models – 0.073983 (bolt_small) vs. 0.074206 (bolt_base) vs. 0.073989 (bolt_mini) vs. 0.074245 (bolt_tiny) for **MAPE**, while Prophet gives 0.074331. Difference scales are similar for **MAE** and **RMSE**.

In Slope Calculation, though fine-tuned Chronos models improve, the conservative prediction issue persists, though the performance does increase a lot after fine-tuning. Chronos gives 0.000986 (Bolt small) vs. 0.000945 (Bolt base) vs. 0.001527 (Bolt mini) vs. 0.001691 (Bolt tiny) for **MAPE**, while Prophet gives 0.000777, but the 0.015% difference is really negligible.

4.3 OpenAI o3-mini

OpenAI released the o3-mini model on Jan 31, 2025, which excels in coding and math problem-solving, especially with its "chain-of-thought" reasoning ability, allowing it to think through and solve things step-by-step. This feature enhances logical reasoning and problem-solving accuracy (OpenAI, 2025)[4].

Compared to earlier work that explored the use of GPT-4o for the same tasks, we want to find out whether o3-mini can deliver improved results. Studies indicate that for tasks requiring five or more reasoning steps, o3-mini and similar models achieve over 16% higher accuracy than GPT-4 (Microsoft, 2025). o3-mini is also cheaper. Its operational costs are about 50% lower compared to GPT-4o (Microsoft, 2025). Given that most engineering tasks involve processing large datasets, o3-mini stands out as a viable option for such applications, trading off between cost, performance, and precision.

To apply the o3-mini model to engineering

tasks, we employed a strict testing process to ensure reliable results. For each task, we started by verifying the ground truth, then connected to the API, experimented with different prompt methods, assessed the model's performance, and repeated the tests multiple times to verify consistency. Since o3-mini does not allow file attachments, we had to reformat raw time series data into formatted text prior to feeding it into the model.

Rough Performance:

After testing approximately 3,000 rows of data, in the threshold exceedance task, o3-mini showed excellent performance, achieving 100% **accuracy** with a simple prompt. However, there was randomness in the results, with accuracy dropping 2 - 3% depending on the random seed. This would imply that for larger datasets, the accuracy might decline further.

For the more complex slope calculation task, o3-mini's performance declined and became highly sensitive to prompt variations. Through systematic experimentation, we identified two key techniques that significantly improved accuracy: (1) explicitly defining each calculation step, and (2) manually selecting numerically stable equations in the prompt to reduce floating-point errors.

A common issue in numerical computations is loss of significance, which can lead to larger discrepancies between the computed slope and the actual value. We found that stating "only consider two decimal places" in the prompt or choosing numerically stable equations manually helped mitigate this issue. With these adjustments, the model produced the correct slope within two-decimal precision in 90% of cases.

Later we intend to examine failure instances to learn when and why o3-mini fails in slope estimation. And to further cross-validate the performance, we plan to try out larger sets of data from varied sources. This will help us ascertain whether the model is uniform across various sets of data distribution or not.

4.4 Deepseek R1

Deepseek is a recently released LLM that demonstrates competitive reasoning ability. Unlike other LLMs, it shows its thinking process before delivering results, which allows users to troubleshoot problems more easily. It is also a cost-efficient pricing model, especially suitable for enterprise and specialized applications compared with OpenAI's GPT models, etc. DeepSeek emphasizes affordability for high-volume, domain-specific tasks, resulting in lower operational expenses, particularly for long-sequence or real-time temporal data processing.

We used prompt - based interactions without fine-tuning to integrate Deepseek into our tasks. The key to effective implementation is structuring prompts in a way that enables Deepseek to interpret the data correctly and execute numerical operations accurately.

For threshold exceedance, we design prompts specifying the target variable (e.g., volt), predefined threshold (e.g., 95th percentile), and time range. The structured prompt is the following:

You are asked to count the number of points exceeding the threshold of x for those y points. Notice the decimal points when comparing. Compare each datapoint one by one. If you could use code to count, feel free to use code.

For slope calculation, prompts requested trend analysis, and the model's output was compared with the ground truth.

For slope calculation, prompts are structured to request trend analysis, and the model's output was compared with the ground truth slope derived from manual linear regression to evaluate performance. The structured prompt is concise and straightforward:

You are asked to calculate the slope of a given time series. If you could use code to

count, feel free to use code.

Rough Performance:

For the threshold exceedance task, we evaluated Deepseek's performance on 300 data points. The model achieved 100% **accuracy** when the dataset size remained within a manageable range. However, one significant drawback was its computational efficiency, taking over 3 minutes per execution.. Additionally, when the number of data points in a single prompt exceeded 200, Deepseek struggled due to computational limitations, leading to incorrect answers and a drop in **accuracy** to 50%.

Prompt design played a crucial role in Deepseek's accuracy. We found that including the term "decimal point" in the prompt was necessary for the model to correctly compare floating-point numbers, otherwise Deepseek failed to recognize numerical order correctly. Plus, instructing the model to use code-based computations significantly improved accuracy.

For the slope calculation task, we conducted a similar evaluation with 300 data points, and the computational limitation issue became even more pronounced. Since the time series variables (e.g., pressure) contained values with four decimal places, Deepseek struggled to compute the slope accurately, leading to substantial discrepancies compared to the ground truth. For instance, in one dataset, the **actual slope** was -8.08, whereas DeepSeek **predicted** -4.16, indicating a clear deviation.

However, after technical optimizations by the Deepseek team, the computational bottleneck was partially alleviated. In a follow-up evaluation, Deepseek's slope calculations closely matched the ground truth, with an error margin as small as 0.001, in some cases only differing due to rounding. Despite achieving near-perfect accuracy, the issue of processing time remained. Compared to the threshold exceedance task, slope calculations took even longer, with each execution averaging over 7 minutes.

These findings highlight DeepSeek's strong numerical reasoning capabilities but also underscore its computational inefficiencies when handling large-scale numerical operations. While accuracy improved significantly with optimized prompts and backend enhancements, further optimizations are required to reduce execution time.

4.5 LangChain Pandas Agent

LangChain Pandas is a specialized agent in the LangChain framework that enables natural-language interaction with pandas DataFrames. The Pandas agent will convert user prompts into Pandas code for execution, relying on an LLM (initially GPT-3.5-turbo in this project) for query interpretation and code generation.

Given its strong performance from last semester's team, we continued to explore and evaluate LangChain Pandas to enhance our understanding of the framework. Additionally, its ability to respond to prompts in plain English aligns closely with the project goal, allowing users to investigate the data without the need to write code. This ability makes data investigation more accessible, efficient, and user-friendly, particularly for non-programmers, making it a powerful tool for data exploration and analysis.

For each task in the project, a LangChain pandas agent is built using an LLM-powered API (such as ChatGPT) along with the relevant data for the task. Once the agent is created, we optimized its performance through prompt selection and customization. First, the LLM generated a set of potential prompts, and we tested each one to choose the best one with the highest accuracy. Then, we further customized the selected prompt to enhance its relevance to our task and to ensure optimal performance. Since the performance of the agent relies significantly on the phrasing of the prompt, leveraging the same LLM that powers the agent ensures consistency and alignment with the API.

Rough Performance:

At this stage, the LangChain Pandas agent has demonstrated strong performance in handling numerical data. It manages to complete both slope calculation and threshold exceedance tasks with an **accuracy** of 100%. To further validate these results, we plan to test the agent on a broader and more diverse dataset later in the project.

However, its performance degraded when processing text data. The performance decline originated from the agent's tendency to produce false positives, which reduced **accuracy** to approximately 90%, likely caused by the token length limitation of GPT-3.5-turbo. We used text splitting (fragment data into chunks) to address this constraint, but it led to information loss and reduced accuracy. Additionally, finding the optimal value for the amount of text in each chunk and the overlap between them are crucial for the text-splitting process, which heavily influences the agent's performance and efficiency. Since this token length limitation is specific to GPT-3.5-turbo, we plan to experiment with other LLMs to reassess the agent's performance without the text length restrictions.

4.6 Benchmark Model

4.6.1 Prophet

Prophet is a popular forecasting model, favored for its simplicity, interpretability, and probabilistic forecasting capabilities. It breaks down time series data into trend, seasonality, and holiday effect components. This decomposition enables it to effectively capture long-term trends, regular cycles, and the impact of special events. By simulating multiple forecast paths based on uncertainty estimates of these components, Prophet generates not only point forecasts but also confidence intervals, similar to what Chronos does. Its intuitive parameterization allows users to easily adjust the model to fit different datasets, making it suitable for a wide range of time-series forecasting tasks, from business demand prediction to environmental data analysis. In this project it's chosen as a baseline model for comparison.

Since Prophet also only supports time series forecasting task, same techniques as Chronos were applied when dealing with the other two tasks. And rough performance for this model is discussed and compared during Section 4.2 together with Chronos.

4.6.2 LLM Benchmark

GPT-4o is one of OpenAI's most popular and widely adopted LLMs, currently leading global usage. It's also used by last semester's team, so we plan to use it for comparison with the remaining 3 LLM models: GPT o3-mini, Deepseek R1, and LangChain Pandas Agent. Since the procedures regarding how to perform the tasks using GPT-4o are similar to GPT o3-mini, we won't list the details here.

5 Conclusion & Next Steps

5.1 Conclusion

After determining the overall goal of the project, i.e., to bypass coding in real world industry workflow and to explore what's going on in the forefront of time series analysis domain, we've conducted a detailed literature review and background research and selected the model accordingly. Following that, we've built a solid understanding of all models that we'll use in this project and illustrated the usage as well as performance for each model on 3 major and commonly used time series tasks. Currently based on the rough performance of our models,

we are on the right track, but we still have a lot of work to do, mainly about testing the generalizability of our models.

5.2 Next Steps

1. For Threshold Exceedance and Slope calculation tasks, we plan to extract 100 subsets (i.e. 100 time series) of data from the PdM telemetry dataset, each with 100 data points. Since telemetry dataset contains 4 variables (voltage, rotation, pressure, and vibration), we plan to extract 25 time series for each variable. Then we'll perform these two tasks on each subset of data, then combine the results for performance evaluation. For Forecasting task, we plan to extract 20 subsets of data from PdM telemetry dataset, and the remaining 80 subsets of data will be chosen from dataset list in the Appendix section. Afterwards we'll also combine the results for better performance evaluation.

2. For Chronos model, currently we've only tried to use CPU during fine-tuning, but later we'll explore if using GPU will improve the performance; also, current data precision is too high, e.g. each data point looks like 176.217853015625 in telemetry dataset, since some models are very sensitive to data precision as illustrated from last semester's team, we'll try to cut the precision to be 2 decimal points and see if the model performance will change; finally, we'll further explore better fine-tuning parameters.

3. For OpenAI o3-mini model, we plan to further explore its potential in time-series forecasting tasks and compare its performance with Prophet and Chronos. Furthermore, since this model currently underperforms in slope calculation task compared to LangChain and DeepSeek, we intend to conduct a deeper analysis of its error patterns and to determine whether improvements can be achieved through prompt engineering and data preprocessing optimization.

4. For Deepseek, we aim to enhance execution efficiency through prompt engineering or model parameter adjustments. Thus far, we have conducted tests on the first two tasks; moving forward, we will evaluate its forecasting capabilities and establish metrics to quantify its performance.

5. For LangChain Pandas Agent, we want to investigate its ability to analyze text data further by experimenting with different APIs and incorporating various text data types. After thoroughly evaluating its performance on text data, we aim to develop an agent that can analyze both text and numerical data. Lastly, to ensure the significance of our results, we intend to build a pipeline procedure to assess the model performance across 50 to 100 datasets, gaining a comprehensive view of how well the model performs under different circumstances.

6. Finally, if time permits, we'll investigate other related LLMs techniques. For example: how to use LLMs to conduct data retrieval using only prompts. To clarify, currently we assume we have the processed data to be analyzed, but there is another step beforehand – to extract the data from the database. Given the power of LLMs models, why not simply providing the structure of database and tasks as context (prompt), and let the LLMs do the remaining steps – generate SQL command and retrieve data itself.

6 Contribution

- **Zhiqi Ma**: Main contributor on introduction section, datasets description section, tasks section, models overview, all parts for Chronos and Prophet, and finding other datasets for test. Team Captain: set up team meetings, milestones, and manage progress.
- **Bingqian Lu**: Main contributor to the LangChain Pandas Agent model, data visualization and text data description, and next step sections.
- **Yangruonan Lin**: Main contributor on Deepseek model and next steps sections.
- **Ruobing Zhang**: Main contributor to the OpenAI o3-mini model and the GPT-4o baseline model, as well as the next step section.

References

[1] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series, 2024.

[2] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G. Wilson. Large language models are zero-shot time series forecasters. arXiv, August 2024. Accessed 12 Dec. 2024.

[3] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-llm: Time series forecasting by reprogramming large language models. arXiv, October 2024. Accepted by the 12th International Conference on Learning Representations (ICLR 2024).

[4] OpenAI. Introducing o3-mini: A new generation of ai for advanced reasoning. January 31 2025. Accessed: 2025-03-06.

[5] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[6] Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts, 2025.

[7] Silin Yang, Dong Wang, Haoqi Zheng, and Ruochun Jin. Timerag: Boosting llm time series forecasting via retrieval-augmented generation. arXiv, December 2024. Cite as: arXiv:2412.16643 [cs.AI].

[8] Tian Zhou, PeiSong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all:power general time series analysis by pretrained lm, 2023.

A Dataset List

Note: datasets with a * were used last semester for this project, so they're chosen here for performance comparison. Since Chronos is a pre-trained foundation model, datasets used during its pre-training process have been excluded.

For comprehensive evaluation, better to find datasets across different time frequencies (and domains). e.g.: 1min, 5min, 30min, 1H, 1D, 1W, 1M, 3M, 1Q, 1Y. Below is a list of datasets possibly to be used for testing.

- 0.5 second:

1. HeartRateDataset – DARTS

- 10 min:

1. Appliances Energy Prediction – UCI ML Repo

- 15 min:

1. ETTm1Dataset – DARTS
2. ETTm2Dataset – DARTS

- 30 min:

1. Aus Electricity Demand – Monash
2. TaylorDataset – DARTS

- 1 H:

1. San Francisco Traffic – Monash
2. ETTh1Dataset – DARTS
3. ETTh2Dataset – DARTS
4. TrafficDataset – DARTS
5. Air Quality – UCI ML Repo

- 1 D:

1. COVID Deaths – Monash
2. Saugeen River Flow – Monash
3. US Births – Monash
4. NN5(Daily)– Monash
5. ExchangeRateDataset – DARTS
6. TemperatureDataset – DARTS

- 1 W:

1. Dominick – Monash
2. NN5(Weekly)– Monash

3. ILINetDataset – DARTS
4. USGasolineDataset – DARTS

- 1 M:

1. MonthlyMilkDataset* – DARTS
2. AirPassengersDataset* – DARTS
3. SunspotsDataset* – DARTS
4. WineDataset* – DARTS
5. GasRateCO2Dataset* – DARTS
6. M1 (Monthly) – Monash
7. Tourism (Monthly) – Monash
8. CIF 2016 – Monash
9. Car Parts – Monash
10. FRED-MD – Monash
11. Hospital – Monash
12. IceCreamHeaterDataset – DARTS

- 1 Q:

1. WoolyDataset* – DARTS
2. AusBeerDataset* – DARTS
3. M1 (Quanterly) – Monash
4. M3 (Quanterly) – Monash
5. M4 (Quarterly) – Monash
6. Tourism (Quarterly) – Monash

- 1 Y:

1. M1 (Yearly) – Monash
2. M3 (Yearly) – Monash
3. M4 (Yearly) – Monash
4. Tourism (Yearly) – Monash