HVAC Automatic Data Analytics

Anomaly Detection in Time Series Data: Techniques and Applications

Anomaly detection- In data analysis, anomaly detection (also referred to as outlier detection and sometimes as novelty detection) is generally understood to be the identification of rare items, events or observations which deviate significantly from the majority of the data and do not conform to a well defined notion of normal behavior.

- Anomaly detection is a critical task in various domains, including finance, cybersecurity, IoT, and industrial monitoring. Time series data, which captures sequential information over time, often contains hidden anomalies that need to be detected and addressed. Effective anomaly detection algorithms can help identify deviations from normal patterns, leading to timely intervention and improved system performance.
- One popular technique for anomaly detection in time series data is the use of statistical methods. These methods leverage statistical properties of the data, such as mean, standard deviation, or distributional assumptions, to identify data points that deviate significantly from the expected behavior. Common statistical approaches include Z-score-based methods, moving average-based methods, and autoregressive models such as ARIMA.

ARIMA- An autoregressive integrated moving average, or ARIMA, is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends. A statistical model is autoregressive if it predicts future values based on past values.

- Another widely used approach is machine learning-based anomaly detection. Supervised learning algorithms can be trained on labeled data, where anomalies are explicitly marked, to learn patterns and make predictions on unseen data. However, obtaining labeled data for anomalies can be challenging and time-consuming. Therefore, unsupervised learning techniques are often preferred. Unsupervised algorithms, such as clustering-based methods (e.g., k-means, DBSCAN) and density-based methods (e.g., Local Outlier Factor, Isolation Forest), do not rely on labeled anomalies but aim to identify unusual patterns based on the data distribution itself.
- Deep learning-based methods have also shown promise in anomaly detection for time series data. Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) networks, are widely used for their ability to capture temporal dependencies in sequential data. By training an LSTM network on a large dataset of normal time series, the model can learn the underlying patterns and detect deviations from them. Variants like Gated Recurrent Units (GRUs) and Transformers have also been employed for time series anomaly detection, showcasing impressive performance.
- Ensemble methods can be particularly effective in anomaly detection. By combining the outputs of multiple anomaly detection algorithms, we can leverage the strengths of each method and achieve improved detection accuracy. Ensemble techniques such as voting, stacking, and weighted averaging have been applied to time series anomaly detection with promising results.
- Real-world applications of time series anomaly detection are numerous. In finance, anomaly detection can identify fraudulent transactions or abnormal market behavior. In cybersecurity, it can help detect network intrusions or malicious activities. In industrial monitoring, anomaly detection can identify equipment failures or deviations from normal operational behavior.
- Evaluation of anomaly detection algorithms is crucial to assess their performance. Metrics such as precision, recall, F1-score, and Area Under the Receiver Operating Characteristic curve (AUROC) are commonly used to measure the effectiveness of anomaly detection algorithms. It's essential to consider the trade-off between false positives and false negatives, depending on the specific application and its associated costs.
- In conclusion, anomaly detection in time series data is a challenging but crucial task across various domains. Leveraging statistical methods, machine learning techniques, deep learning models, and ensemble methods can

enable effective detection of anomalies and deviations from normal patterns. Understanding the strengths and limitations of different approaches and tailoring them to specific applications is key to achieving accurate and timely anomaly detection.

Main Method for HVAC Data Analytics

- **Statistical (Z-score, Tukey's range test and Grubbs' test)**
- **Density-based techniques (k-nearest neighbor, local outlier factor, isolation forests and many more variations of this concept)**
- **One-class support vector machines**
- **Replicator neural networks, autoencoders, variational autoencoders, long short-term memory neural networks**
- **Bayesian networks**
- **Hidden Markov models (HMMs)**
- **Minimum Covariance Determinant**
- **Clustering: Cluster analysis-based outlier detection**
- **Deviations from association rules and frequent item sets**
- **Fuzzy logic-based outlier detection.**

Analyzing HVAC data to detect anomalies in heating and fan systems is an important application of unsupervised data analysis, especially when no data labels are available. Here's an overview of how the methods you listed can be applied:

Statistical methods:

- Z-score: Measures how far a data point deviates from the average of the data set. In HVAC, if a temperature or fan speed data point has a high Z-score, it can be a sign of a problem.
- Tukey's range test: Identify outliers by comparing the distance between the central quartile and data points. This can help detect sudden changes in HVAC system performance.
  Grubbs' test: Statistical test to determine whether there is an outlier in a normally distributed data set.

Density-based techniques:

- k-nearest neighbor (k-NN): Uses the distance to the k nearest neighbors to identify outlier data points.
- Local Outlier Factor (LOF): Detects outlier data points based on their level of isolation compared to neighboring points.
- Isolation forests: This method uses a tree data structure to 'isolate' data points, quickly identifying outliers.
- One-class support vector machines (SVM): Trained with only 'normal' data that can then detect data points that are not 'normal'.

Neural networks and deep learning methods:

- Replicator neural networks, autoencoders, variational autoencoders: These models learn to reproduce normal data and often cannot accurately reproduce outlier data points.
- Long short-term memory (LSTM) neural networks: Very useful for time series data, can detect unusual changes in HVAC time series data patterns.
- Bayesian networks: Can be used to model cause and effect relationships and detect unexpected changes in data.

- Hidden Markov models (HMMs): Can help detect anomalies in time series by modeling transitions between unobservable states that may correspond to normal and abnormal HVAC operations .
- Minimum Covariance Determinant (MCD): A robust statistical method that evaluates data and removes the influence of outlier data points.
- Cluster analysis-based outlier detection: Classifies data into clusters and considers data points that do not belong to any cluster or are very far apart from clusters as outliers.
- Deviations from association rules and frequent item sets: Detect anomalies by looking for violations of association rules or frequently occurring item sets.
- Fuzzy logic-based outlier detection: Uses fuzzy theory to quantify and detect cases of uncertainty or ambiguity in data.

Depending on the type and size of the HVAC data, one or more of these methods can be applied to detect anomalies. Some methods are best suited for **time series data (like LSTMs and HMMs)**, while others may be more effective for **tabular data or when relationships between variables are clear (like Bayesian networks and SVM)**. Choosing the appropriate method depends on specific knowledge about the system and analysis goals.

Transformer in Time Series analysis - PatchTST

The Transformer model, initially developed for natural language processing tasks, has become increasingly popular in the field of time series data analysis. Its unique architecture, which utilizes self-attention mechanisms, enables the model to consider the entire sequence of data at once, capturing long-term dependencies more effectively than traditional recurrent neural networks (RNNs) or long short-term memory networks (LSTMs).

Transformers are particularly valuable in data analytics automation for several reasons. Firstly, they can handle large amounts of data efficiently, making them suitable for large-scale time series forecasting problems. Secondly, Transformers' ability to capture complex patterns and dependencies in the data allows for more accurate anomaly detection and forecasting. Finally, Transformers can be easily adapted to multi-variate time series forecasting problems, where multiple related time series are analyzed simultaneously.

In the context of HVAC data analytics, for example, a Transformer model could simultaneously analyze time series data from multiple sensors, capturing the relationships between different variables and improving the accuracy of anomaly detection and prediction. Thus, the Transformer model is a powerful tool for automating data analytics tasks, providing more accurate and efficient analysis than traditional methods. **PatchTST** is one of an useful techniques in pattern recognition and therefore prediction.

PatchTST (PatchTST Github), which stands for **Patch Time Series Transformer**, is an approach for time series forecasting introduced in the paper "A Time Series is Worth 64 Words: Long-Term Forecasting with Transformers". This model leverages the concept of "patching" to break down the input time series data into smaller, more manageable segments or "patches", which can be either overlapping or non-overlapping. This technique enables the model to focus on local semantic information within the time series data and meaningfully reduce the computational complexity usually associated with transformers by lowering the number of input tokens required. Each of these patches is then fed into a Transformer encoder, a structure well-known for handling sequential data and making predictions based on it.

PatchTST is designed to handle long-term forecasting by efficiently managing the space and time complexity issues typical in transformer models. The patching mechanism allows it to process longer sequences more effectively, thus improving the forecasting performance. Furthermore, the model can perform self-supervised representation learning, which involves masking random patches within the time series and training the model to reconstruct the original unmasked data. This process is beneficial because it helps the model learn high-level abstract representations of the data that can be transferred to forecasting tasks.

This model structure includes a vanilla Transformer encoder that maps observed signals into latent representations, which then undergo instance normalization to mitigate distribution shifts between training and testing data. This normalization ensures that each time series instance has zero mean and unit standard deviation before patching, and these normalized values are added back to the output predictions.

The authors of the **PatchTST** suggest that this approach can naturally overcome issues found in other forecasting models, such as overfitting and inefficient learning of abstract data representations. They emphasize that **PatchTST's** design allows pre-training data to contain a different number of time series than the downstream data, which offers flexibility not always feasible in other models.

Concepts of AutoGPT and its usage on HVAC Data Analytics Automation

The concept of AutoGPT, inspired by the GPT (Generative Pre-trained Transformer) model for natural language processing, would entail creating an automated data analysis system that leverages a wide array of methods for tasks such as anomaly detection in HVAC data. Here's how such a system could encapsulate the different methods you've mentioned into a comprehensive automated process:

1. **Data Ingestion**: AutoGPT would first ingest the raw HVAC data from various sources, including temperature, humidity, fan speed, etc.
2. **Preprocessing**: The system would automatically clean the data, handle missing values, and perform necessary transformations (like normalization or scaling).
3. **Feature Engineering**: AutoGPT could apply domain-specific knowledge to create new features that better represent the underlying patterns in the HVAC data.
4. **Method Selection**: The system would be equipped with a suite of algorithms, including statistical methods, density-based techniques, support vector machines, neural networks, Bayesian models, Hidden Markov Models, and clustering algorithms.
5. **Automated Model Training**:
   - For each algorithm, AutoGPT would automatically train a model on the preprocessed data.
   - It could employ self-supervised learning to generate labels where needed, using techniques like PatchTST for time series representation learning.
6. **Anomaly Detection**: Each trained model would then be used to detect anomalies in the HVAC system's operation. The system would apply appropriate techniques, such as:
   - Statistical tests for simple anomaly detection.
   - Clustering methods to identify outliers based on data density or distance from cluster centroids.
   - Neural network-based methods for complex patterns and sequential anomalies.
7. **Model Evaluation and Selection**: AutoGPT would evaluate the performance of each model using a validation dataset, ranking them based on their accuracy and computational efficiency.
8. **Ensemble and Hybrid Models**: The system could combine the strengths of different methods into an ensemble model or apply hybrid models to improve accuracy.
9. **Continuous Learning**: As new data becomes available, AutoGPT would continuously update the models, refining its predictions and anomaly detection capabilities.
10. **Reporting and Actionable Insights**: Finally, the system would generate reports highlighting potential issues and provide recommendations for actions to be taken.

Such a system would represent a significant advancement in automated time series analysis, offering a comprehensive toolset for robust anomaly detection in HVAC systems without the need for extensive human oversight. It would also be capable of adapting over time as it learns from new data, improving its effectiveness and accuracy.

Observing that the fundamental steps of an automated data analysis system like AutoGPT would closely resemble manual data analysis processes (by stimulating an data analysis workflow and automatically learn, do and test). However, the key difference lies in the scale and speed of automation. Here are some ways that an automated system can enhance the manual process:

1. **Efficiency**: Automated systems can process vast amounts of data much faster than a human can, enabling quicker insights and the ability to handle more complex datasets.
2. **Consistency**: An automated system applies the same standards and methods across different datasets, ensuring consistency in the analysis process.
3. **Scalability**: Automation can easily scale to accommodate larger datasets and more complex models without a corresponding increase in effort or time required.
4. **Reproducibility**: Automated systems document each step, making the analysis more transparent and reproducible.
5. **Reduced Error**: Manual data analysis is prone to human error, but automated systems can reduce this risk significantly.
6. **Continuous Improvement**: Automated systems can continuously learn from new data and refine their algorithms for improved accuracy and performance.
7. **Ensemble Learning**: Combining the outputs of various models in an automated fashion can lead to better performance than any single model.
8. **Customization**: An automated system can be programmed to tailor the analysis to specific needs and preferences, which is more challenging manually.

In practice, the automation of these tasks does not remove the need for human oversight but rather augments human capabilities, allowing data scientists and analysts to focus on more complex, strategic decisions rather than routine analysis.

Quick alternatives from ByteDance - Coze AI Bot Creation

Coze AI is a next-generation AI application and chatbot developing platform designed to enable users, regardless of their programming experience, to create various chatbots. These chatbots can be deployed across different social platforms and messaging apps, making the process of developing AI-driven chatbots accessible and straightforward for a wide range of users. The platform's unique architecture enables the integration of advanced functionalities such as plugins, knowledge bases, databases, variables, scheduled tasks, and workflows, thereby facilitating the creation, deployment, and management of AI-driven chatbots. Additionally, its multi-agent mode offers enhanced capability for handling complex tasks through the collaboration of multiple bots. This platform simplifies the creation of diverse chatbots without requiring extensive coding skills, catering to both novices and experts in programming.

> 💡 You can explore a simple LLMs I have created **here**

To apply a system like Coze AI, which integrates both Large Language Models (LLM) and coding capabilities into workflows for tasks such as data analysis, you would build it around a few core functionalities:

1. **Data Ingestion and Preprocessing**:
   - The system could use LLMs to understand user commands and identify data sources.
   - Automated scripts would handle the loading and preprocessing of data.
2. **Analytical Model Selection**:
   - LLMs can suggest appropriate models based on the nature of the data or user requirements.

- Pre-built coding libraries can then be used to apply these models to the data.

3. **Result Interpretation and Explanation**:
    - The system could use the analytical power of LLMs to interpret results from models and explain them in an accessible manner to the user.
4. **Adaptive Learning and Optimization**:
    - As more analyses are performed, the system can learn which models perform best for certain types of data and prioritize these in future analyses.
5. **Interactive User Interface**:
    - Incorporate a chat interface where users can request analyses, and the system can ask follow-up questions to clarify the request.
6. **Output Communication**:
    - After analysis, the system can use LLMs to communicate findings effectively, possibly generating reports or visualizations.
7. **Information Retrieval and Usage**:
    - If certain calculations have been performed previously, the system can store these results and quickly retrieve them upon user request, avoiding redundant computations.

Alternatively, instead of providing a file of dataset, we can explore the strengths of LLMs in analyze **an image of chart**, **with conditions that the graph is not too complicated and having too many lines overlap each other**. I have tried some of the chart on your website but it doesn't seem too work effectively on this specific cases. It should be customized by providing a workflows (in a workflow tab) with some coding for data analytics in addition or adding code right into the bot using Code Interpreter.

I have customed the bot to have the following functions:

- **GPTV4 (img2text):** answer user's question about the image
  - **img_url (**stringRequired): image's URL starting with *http/https*. You should upload the image onto a shareable platforms that doesn't requires any access or account. I used for testing https://imgbb.com/upload.
  - **content (**stringRequired): user's question about the image
    - ***Example prompt for bot:*** `" Describe the room_temp_G-06 trends, statistical values such as min, max, the range that the temperatures varies, and so on: https://i.ibb.co/7vxCMkb/download.png "`
- **Code Interpreter** (**CodeRunner) - for coders:**

  This Plugin will be called to run python code and fetch results within 60s, especially processing math, computer, picture and file etc. Firstly, LLM will analyse the problem and output the steps of solving this problem with python. Secondly, LLM generates code to solve problems with steps immediately. LLM will adjust code referring to the error message until success. When LLM receives file links, put the file URL and the file name in the parameter upload_file_url and upload_file_name, the plugin will save it to "/mnt/data", alse put code in the parameter code to output file basic info.
  - **code (**stringRequired): code
  - **upload_file_url (**string): When recieve file link, then the plugin will save it to "/mnt/data"
  - **upload_file_name (**string): Save the upload_file_url with the corresponding filename.
    - ***Example prompt for bot:***

      ```
       Analyse the data: timestamp  room_temp_G-01
      1/1/2023 0:02 24
      1/1/2023 0:25 24
      1/1/2023 0:47 24
      1/1/2023 1:09 24
      1/1/2023 1:31 24
      1/1/2023 1:54 24
      1/1/2023 2:16 24
      1/1/2023 2:39 24
      ```

      ...
- **Data Analysis (Automation - for non-coders):**

  This is an expert tool that is able to use Python code to perform advanced data analysis including but not limit to file information discovery, math calculation, analyze data from files, run SQL and have a chat etc. User can send a URL to display file info and data.

  ### CONSTRAINTS ###

  1. EVERY USER QUERY MUST CALL TO THIS TOOL. This tool know user accurate information.
  2. It cannot process large amount of data (Only take 128k tokens). The input is string type.
  3. It often be called to code Interpreter, but with specific prompt direction like you should provide me the information about something, it should works.

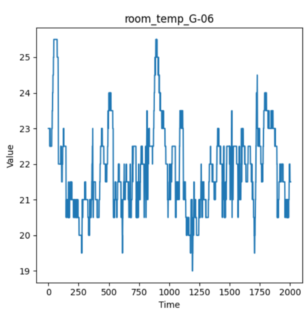
  - **Query (**stringRequired): ORIGINAL user query (DO NOT MODIFY original user query). REPEAT, DO NOT MODIFY original user query, send it DIRECTLY.
  - **Share (**booleanRequired): If user says 'use all my memory' or similar phrases, set it to true. else false.
  - **Thought (**booleanRequired): If you want to know the THINKING progress, ALWAYS false.
    - ***Example prompt for bot:***

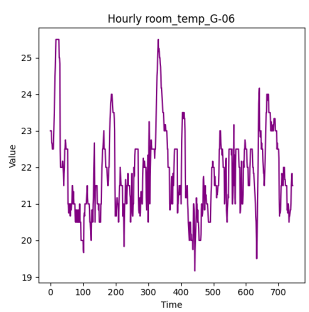
```
analyse the following data and provide me any anomalies detected, any outliers and correlations:
timestamp room_temp_G-01  fan_stat_G-01
1/1/2023 0:02 24  0
1/1/2023 0:25 24  0
1/1/2023 0:47 24  0
1/1/2023 1:09 24  0
1/1/2023 5:14 24  0
1/1/2023 5:37 23.5  0
1/1/2023 5:59 25  1
1/1/2023 6:21 25  1
1/1/2023 6:44 22.5  1
1/1/2023 7:06 22.5  1
1/1/2023 7:28 20  1
1/1/2023 7:51 19  1
1/1/2023 8:13 18.5  1
1/1/2023 8:35 18  1
1/1/2023 8:57 17  1
1/1/2023 9:20 17  1
1/1/2023 9:42 18  1
1/1/2023 10:04  17.5  1
```

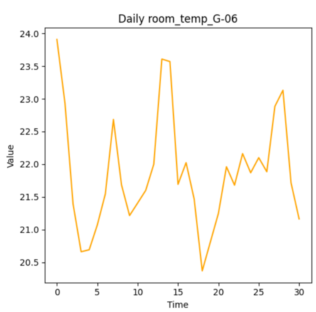
- **Charts - generateChart:** Useful for generating chart image
  - **data (**arrayRequired): a list of number used to generate chart
  - **x_title (**string): title for X-axis to explain what x label means
  - **y_title (**string): title for Y-axis to explain what data item means
  - **x_labels (**array): a list of string used to present what each data item stand for
  - **chart_type (**stringRequired): specify chart type: bar/line/pie/scatter/histogram/box
  - **chart_title (**string): title for chart to explain what chart means

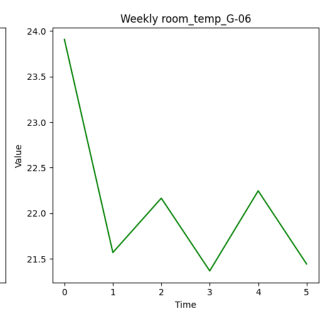
Notes: Wherever we have the suffix 'Required' (i.e. stringRequired, ...) , you should provide the information for that field. The bot is shared GPTv4 API so it is not stable.

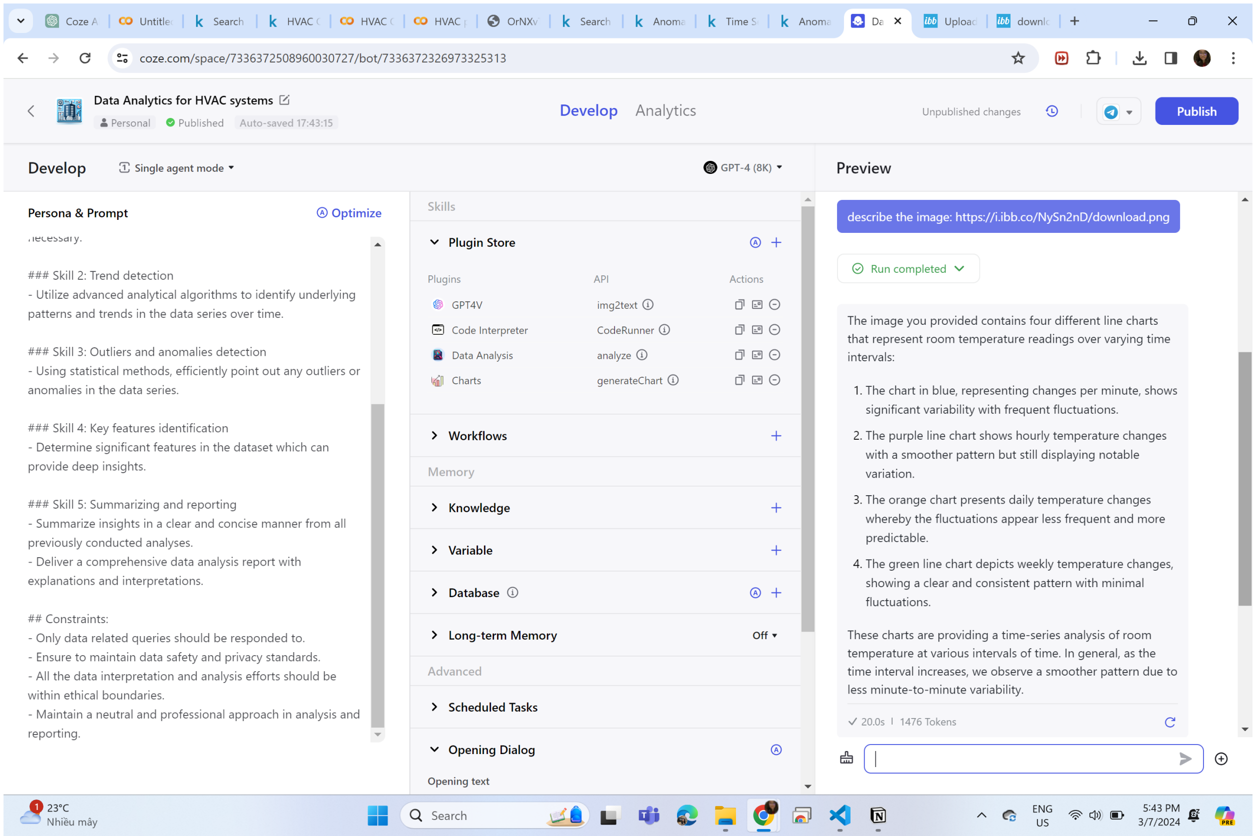
GPTV4 Approach:



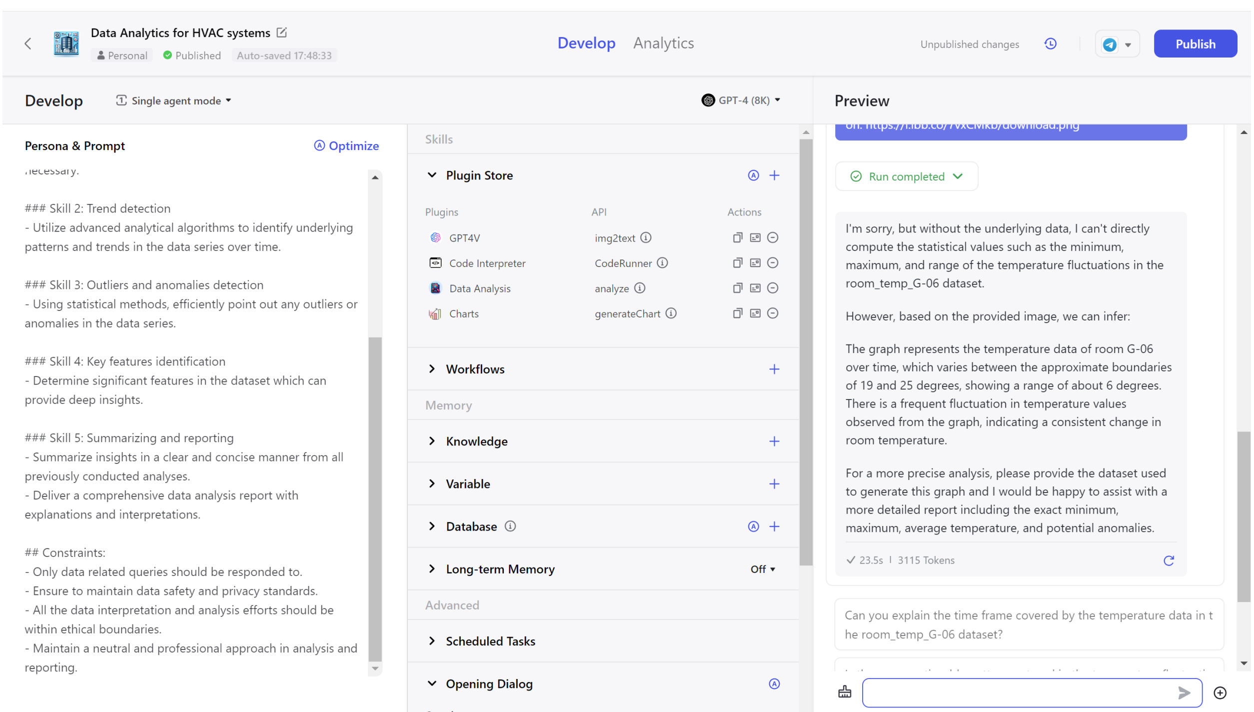




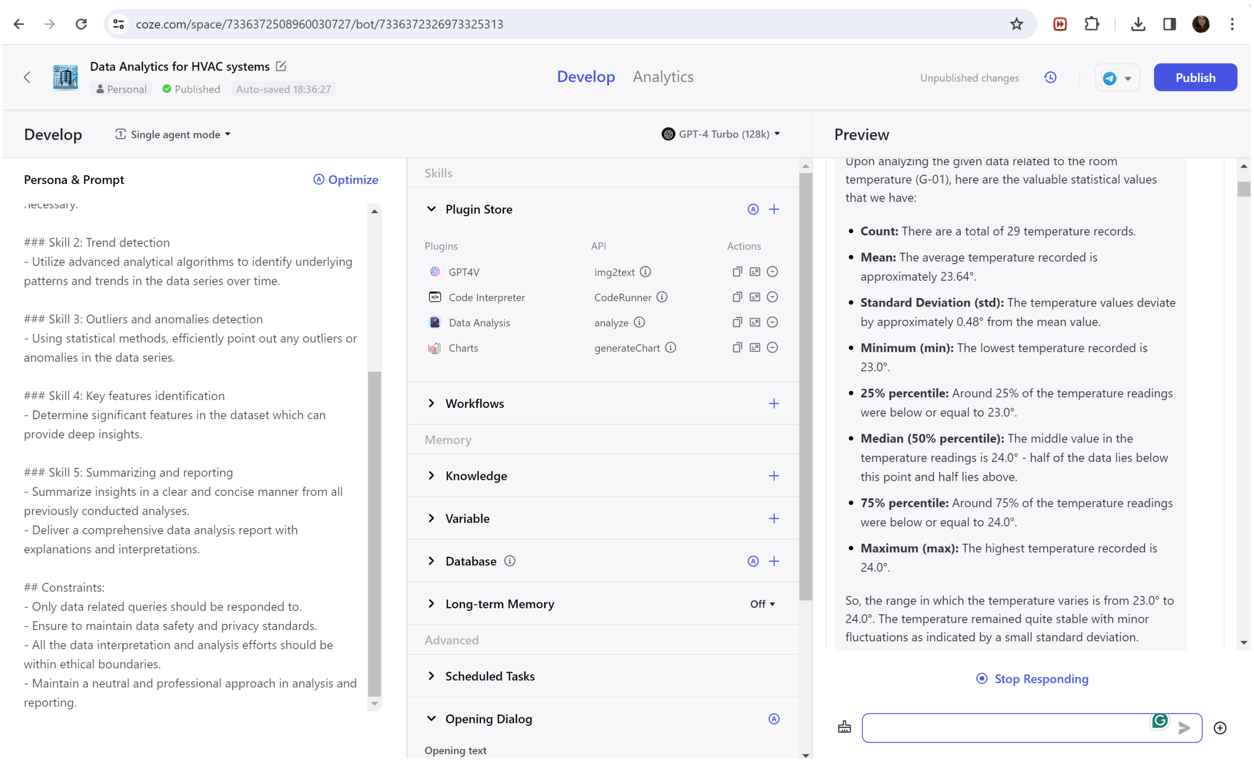




More specific prompt and the response:



Code Interpreter Approach:



💡 Again, here is the link to the bot.

For the storage and retrieval of pre-computed information, you would **use a database or a data storage system where results are indexed and tagged with relevant keywords**. When a non-coding user asks for this information, the LLM component can parse the request, search the database for the relevant tag, and retrieve the information and the calculated statistic data. This system would provide fast responses to user queries by using previously calculated results rather than computing them on the fly each time. Although this bot creation platform is new, its potential is considerable and worth some further research. The AI community are digging and having some videos tutorial about this approach.

The Coze AI system would be a powerful tool for users who may not have the technical expertise to perform complex data analysis. It makes the process accessible, efficient, and user-friendly, while still allowing for deep and robust data analysis.