Simulating dataset

The script simulates Kubernetes metrics data, including CPU usage, memory usage, network I/O, pod status, node failures, and disk usage.It uses libraries like NumPy, pandas, and random for data generation and manipulation.A random seed is set for reproducibility, ensuring consistent results across runs.The script generates 1000 samples with timestamps at 1-minute intervals starting from January 1, 2025.Metrics like CPU, memory, and disk usage are simulated using normal distributions and clipped to realistic ranges.Pod statuses and node failures are randomly assigned based on specified probabilities.Anomalies are introduced by randomly spiking resource usage and marking nodes as failed in 20 instances.The data is combined into a pandas DataFrame and saved as a CSV file named "simulated_k8s_metrics.csv".The script confirms the successful creation and saving of the dataset. This simulated dataset can be used for testing and analyzing Kubernetes system performance and failure scenarios.

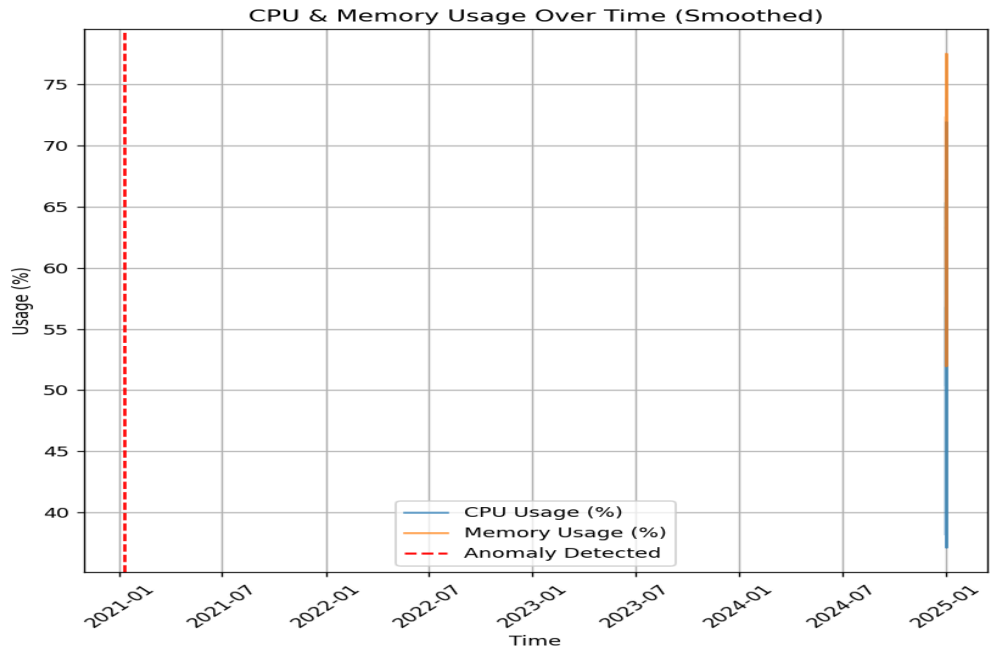
Explore dataset

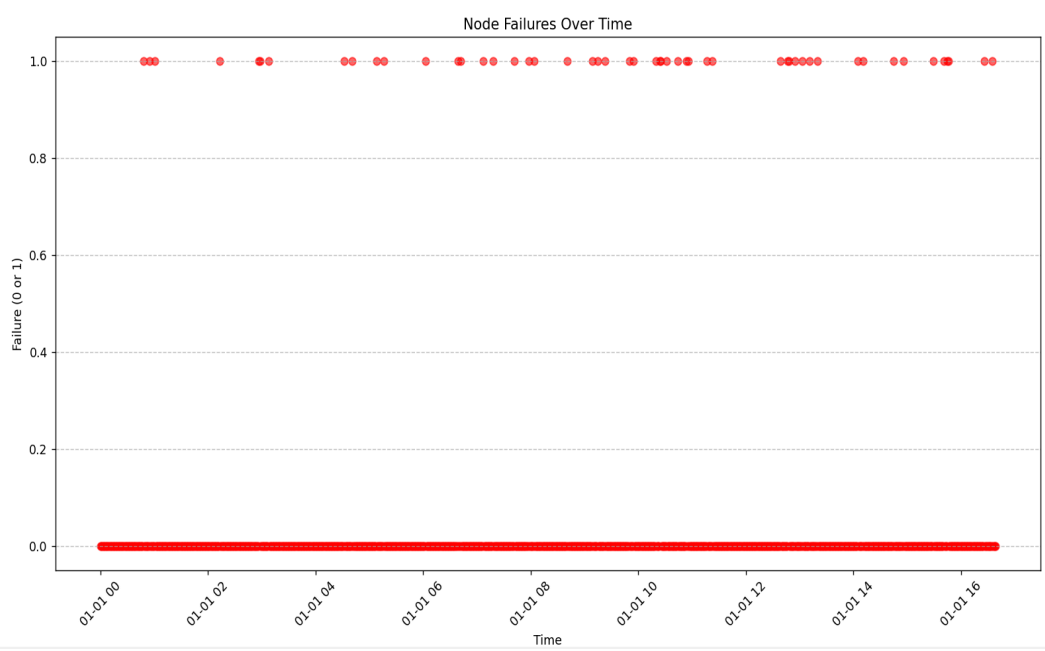
```
PS C:\Users\cathe\OneDrive\Desktop\Kubecoders\scripts> python explore_k8s_data.py
             timestamp  cpu_usage  memory_usage  network_io  pod_status  node_failure  disk_usage
0  2025-01-01 00:00:00  57.450712     73.993554  166.241086           0             0   78.089193
1  2025-01-01 00:01:00  47.926035     69.246337  192.774066           2             0   64.155447
2  2025-01-01 00:02:00  59.715328     60.596304  160.379004           0             0   62.506423
3  2025-01-01 00:03:00  72.845448     53.530632  184.601924           2             0   67.784634
4  2025-01-01 00:04:00  46.487699     66.982233  105.319267           0             0   58.230379
         cpu_usage  memory_usage   network_io   pod_status  node_failure   disk_usage
count  1000.000000   1000.000000  1000.000000  1000.000000   1000.000000  1000.000000
mean     51.392925     61.817904   203.744895     0.198000      0.051000    69.829409
std      17.519970     12.826254    54.840306     0.501046      0.220108     9.862774
min       1.380990     30.596114    49.024392     0.000000      0.000000    38.232962
25%      40.286145     54.224863   169.033737     0.000000      0.000000    63.431399
50%      50.690063     60.842918   201.012579     0.000000      0.000000    69.815998
75%      60.244755     67.733643   234.285521     0.000000      0.000000    76.566827
max     172.791520    170.365826   562.527600     2.000000      1.000000   100.000000
timestamp       0
cpu_usage       0
memory_usage    0
network_io      0
pod_status      0
node_failure    0
disk_usage      0
dtype: int64
PS C:\Users\cathe\OneDrive\Desktop\Kubecoders\scripts>
```

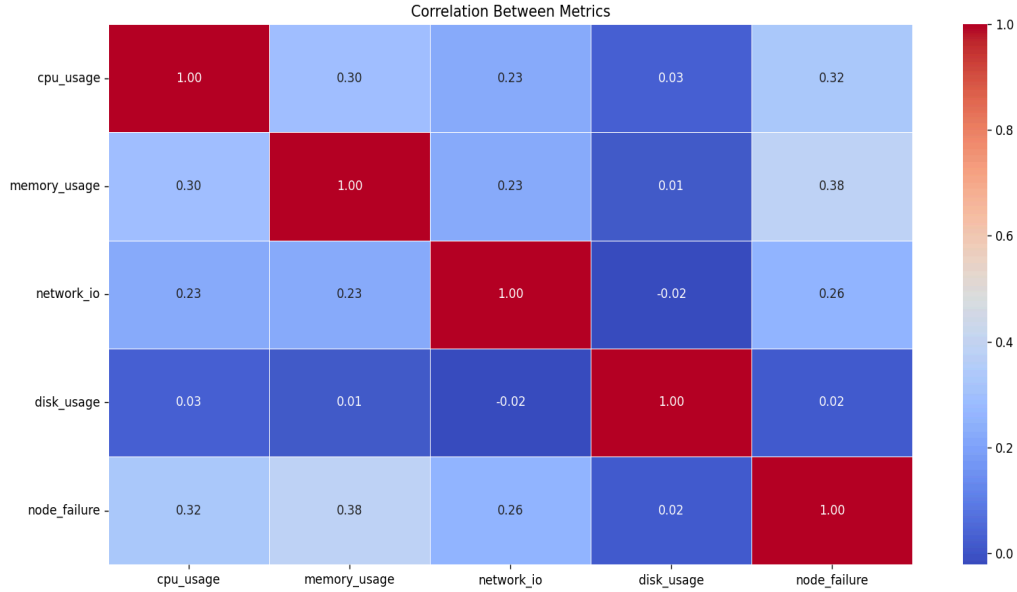
Visualisation of dataset

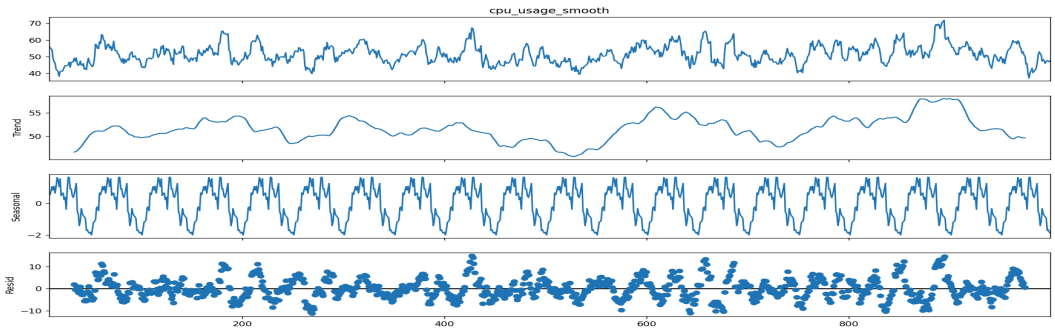
The script processes and analyzes simulated Kubernetes metrics data, including CPU usage, memory usage, network I/O, and node failures. It uses libraries like pandas, matplotlib, seaborn, and scikit-learn for data manipulation, visualization, and anomaly detection. The data is loaded from a CSV file, and timestamps are converted to datetime format for time-based analysis. Rolling means are calculated for CPU and memory usage to smooth the data and highlight trends. A function plots

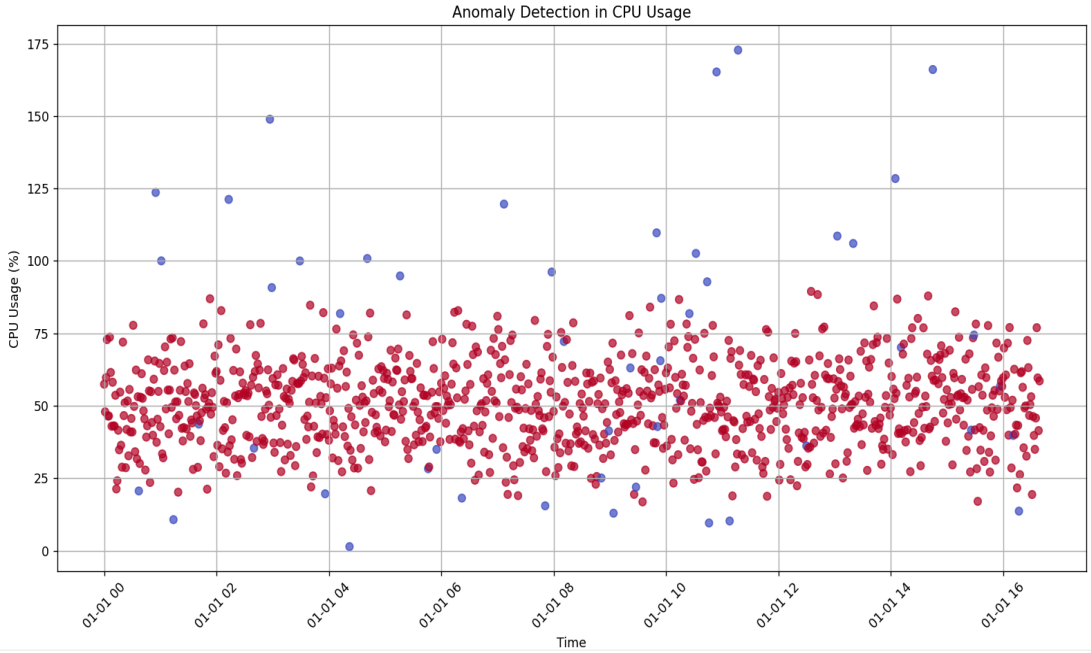
smoothed CPU and memory usage over time, highlighting anomalies with vertical lines. Node failures are visualized using a scatter plot to show failure occurrences over time. A correlation heatmap is generated to show relationships between metrics like CPU usage, memory usage, and disk usage. Seasonal decomposition is applied to CPU usage to identify trends, seasonality, and residuals. Anomalies are detected using the Isolation Forest algorithm and visualized in a scatter plot. The script runs all visualization functions sequentially to provide a comprehensive analysis of the Kubernetes metrics data.





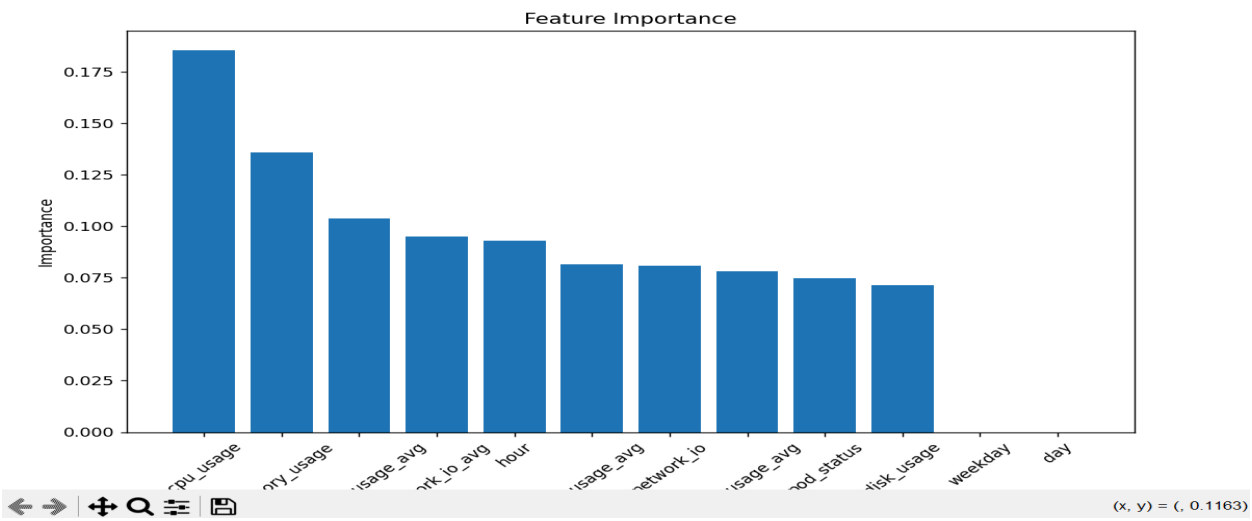

Correlation Between Metrics
cpu_usage
memory_usage
network_io
disk_usage
node_failure
1.00 0.30 0.23 0.03 0.32
0.30 1.00 0.23 0.01 0.38
0.23 0.23 1.00 -0.02 0.26
0.03 0.01 -0.02 1.00 0.02
0.32 0.38 0.26 0.02 1.00
1.0
0.8
0.6
0.4
0.2
0.0


cpu_usage_smooth
Trend
Seasonal
Resid
200
400
600
800


Anomaly Detection in CPU Usage
CPU Usage (%)
Time
01-01 00
01-01 02
01-01 04
01-01 06
01-01 08
01-01 10
01-01 12
01-01 14
01-01 16

Training the model

The dataset (`simulated_k8s_metrics.csv`) is loaded, containing Kubernetes metrics like CPU usage, memory usage, and node failure status. Timestamps are converted to datetime format, and time-based features (hour, day, weekday) are extracted for time-series analysis. Rolling averages for CPU, memory, network I/O, and disk usage are calculated to smooth out short-term fluctuations. The `node_failure` column is converted into binary labels (0 or 1) based on the median value to classify failures. Features are normalized using `MinMaxScaler` to ensure consistent scaling for model training. SMOTE is applied to balance the dataset by generating synthetic samples for the minority class (node failures). The dataset is split into training and testing sets, with 80% used for training and 20% for testing. Hyperparameter tuning is performed using `GridSearchCV` to optimize the XGBoost model for accuracy. The model is trained, evaluated using metrics like accuracy, F1-score, and ROC-AUC, and feature importance is visualized. The trained model and scaler are saved as `.pkl` files for future use in predictions or deployment.




```
📊 Model Performance:
✅ Accuracy: 0.9711
✅ F1-score: 0.9714
✅ ROC-AUC Score: 0.9950
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       190
           1       0.96      0.98      0.97       190

    accuracy                           0.97       380
   macro avg       0.97      0.97      0.97       380
weighted avg       0.97      0.97      0.97       380

✅ Model saved as k8s_issue_model.pkl
✅ Scaler saved as scaler.pkl
PS C:\Users\cathe\OneDrive\Desktop\Kubecoders\scripts>
```

Flask API for Kubernetes Failure Prediction Using Prometheus Metrics

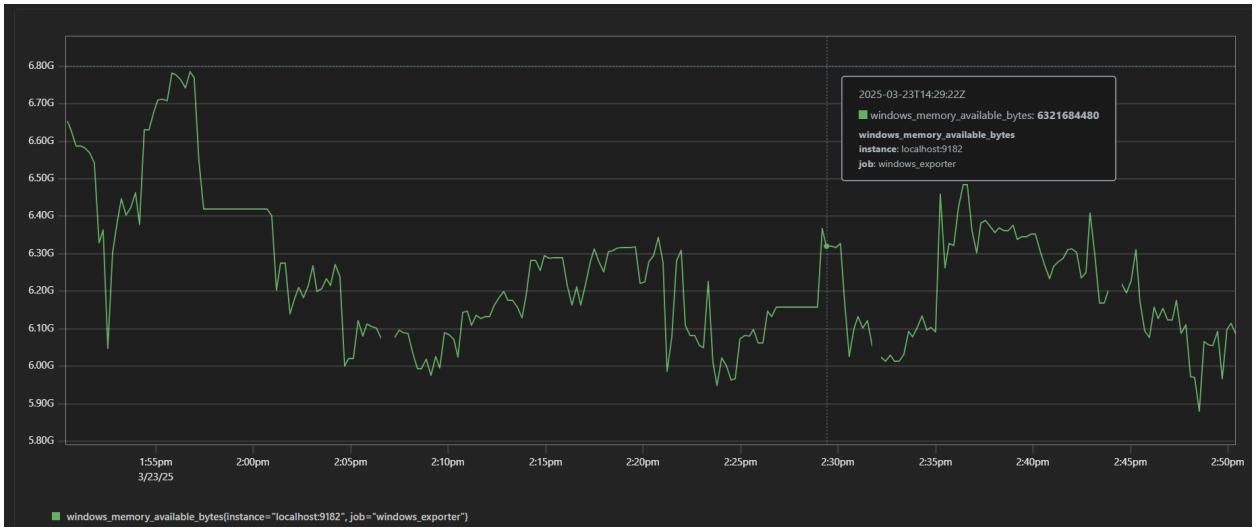
The code sets up a Flask-based API to predict Kubernetes failures using Prometheus metrics. It imports necessary libraries like `requests`, `numpy`, `joblib`, and `Flask` for data fetching, processing, and API creation. A pre-trained machine learning model (`k8s_issue_model.pkl`) and scaler (`scaler.pkl`) are loaded using `joblib`. The API fetches Prometheus metrics (CPU, memory, disk, and network) using the Prometheus query API. Metrics are validated and processed, with default values set to `[0]` if data fetching fails. The `/` route provides a health check to confirm the API is running. The `/predict` route fetches metrics, combines the first two values from each metric, and scales the data for prediction. The model predicts if a failure will occur, returning "Yes" or "No" as the result. Error handling ensures the API returns meaningful messages if data fetching, scaling, or prediction fails. The API runs on `0.0.0.0:5000` and serves as a microservice for real-time Kubernetes failure prediction.

Queries output graph in prometheus url

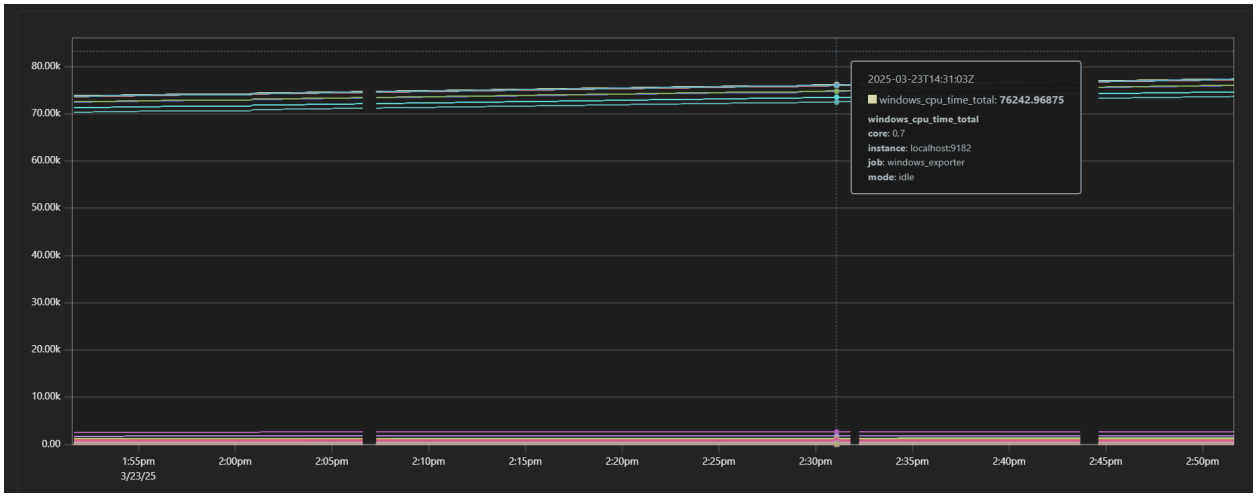
windows_net_bytes_received_total

windows_logical_disk_write_bytes_total

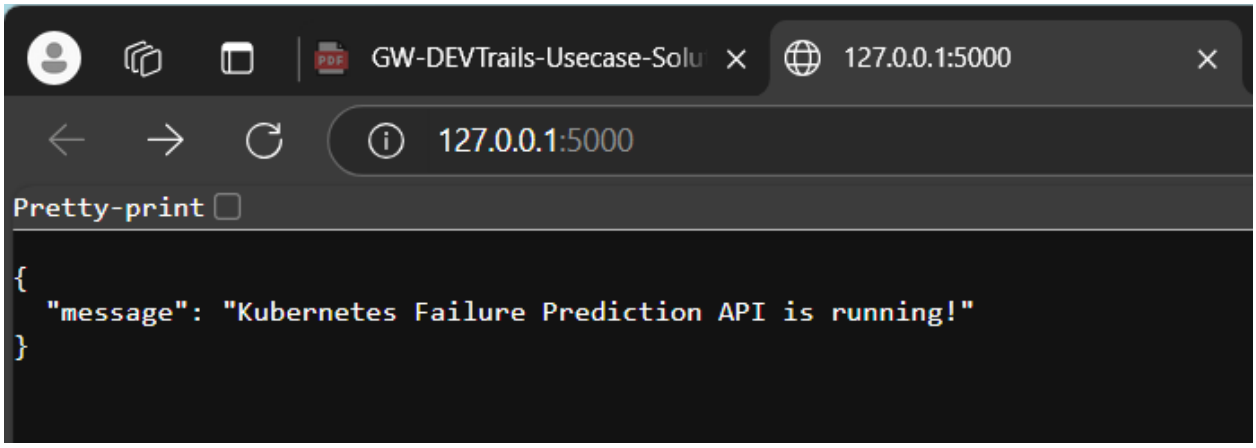
Windows_memory_available_bytes

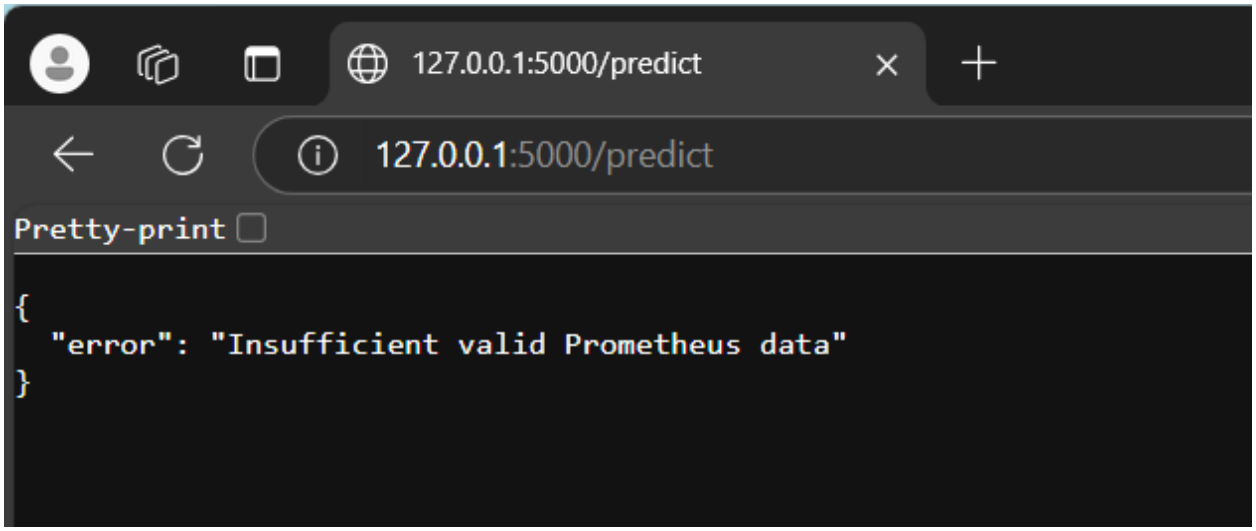


Windows_cpu_time_total

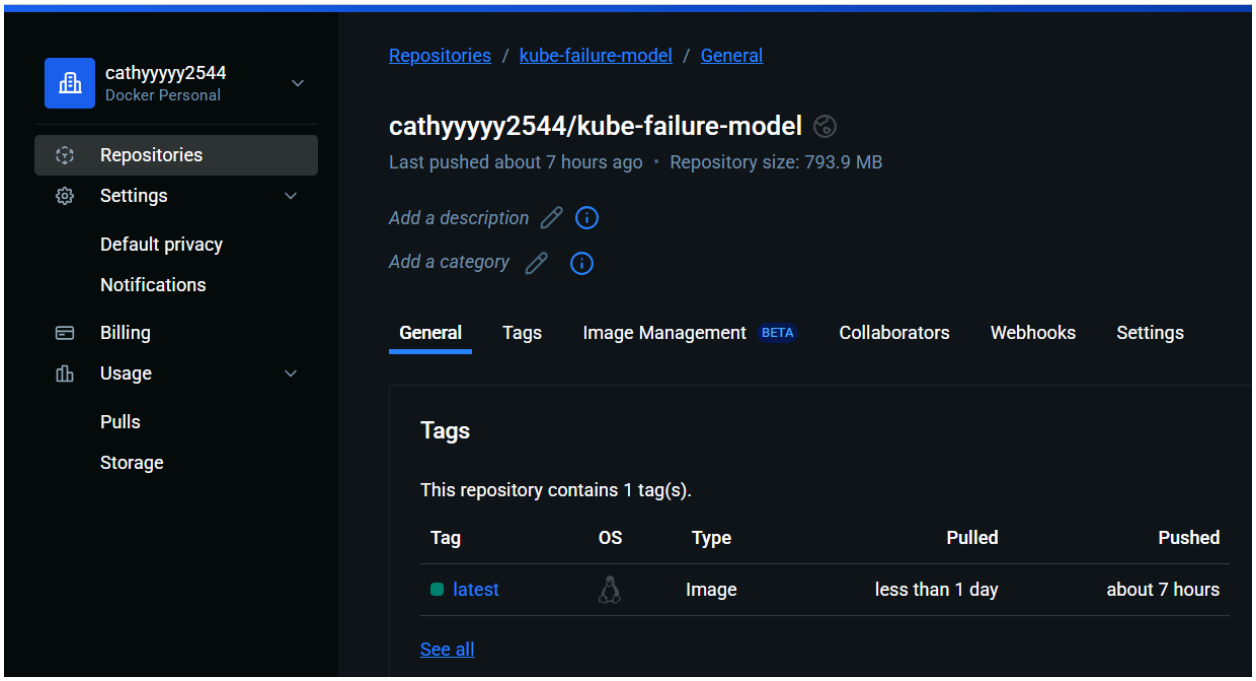


Output (We are working on the validation of the prediction)





DockerHub


cathyyyyy2544
Docker Personal
Repositories
Settings
Default privacy
Notifications
Billing
Usage
Pulls
Storage
Repositories / kube-failure-model / General
cathyyyyy2544/kube-failure-model
Last pushed about 7 hours ago • Repository size: 793.9 MB
Add a description
Add a category
General
Tags
Image Management BETA
Collaborators
Webhooks
Settings
Tags
This repository contains 1 tag(s).
Tag
OS
Type
Pulled
Pushed
latest
Image
less than 1 day
about 7 hours
See all