IT OPERATION AND NETWORK MANAGEMENT

Overview

This documentation covers the platform's architecture, analysis algorithms, and user interface design for an AI-based IT operations and network management system. The platform uses Python, machine learning, and system monitoring tools to provide predictive analytics, asset management, and network monitoring.

Platform Architecture

Components

1. Data Layer
    - Synthetic Data Generation: Generates synthetic data to simulate IT asset performance metrics.
    - Data Preparation: Splits the data into training and testing sets for model training.
2. Analytics Layer
    - Predictive Analytics: Uses machine learning algorithms to predict asset failures.
    - Asset Management: Monitors CPU and memory usage of IT assets.
    - Network Monitoring: Tracks network I/O (input/output) to monitor data transmission.
3. User Interface Layer
    - Visualization: Uses Seaborn and Matplotlib for data visualization.
    - Output Display: Prints evaluation metrics and system usage statistics to the console.

Analysis Algorithms

Synthetic Data Generation

The platform generates a synthetic dataset with the following attributes:

- cpu_usage: Random integers representing CPU usage percentage.
- memory_usage: Random integers representing memory usage percentage.
- network_io: Random integers representing network I/O in bytes.
- asset_age: Random integers representing the age of the asset in years.
- failure: Binary labels indicating whether an asset has failed (10% failure rate).

Predictive Analytics

A RandomForestClassifier is used for predicting asset failures. The model is trained on the synthetic dataset, and its performance is evaluated using confusion matrix and classification report.

Steps:

1. Prepare data by splitting it into features (X) and target (y).
2. Split the data into training and testing sets.
3. Train the RandomForest model on the training set.
4. Evaluate the model on the testing set.

Asset Management

The platform monitors CPU and memory usage using the psutil library.

Steps:

1. Use psutil.cpu_percent() to get CPU usage.
2. Use psutil.virtual_memory() to get memory usage.

Network Monitoring

The platform monitors network I/O using the psutil library.

Steps:

1. Use psutil.net_io_counters() to get bytes sent and received.
2. Monitor network I/O over a period and print the results.

User Interface Design

Visualization

The platform uses Seaborn and Matplotlib for visualizing the dataset. Pair plots are generated to understand the relationships between different features in the dataset.

CODE

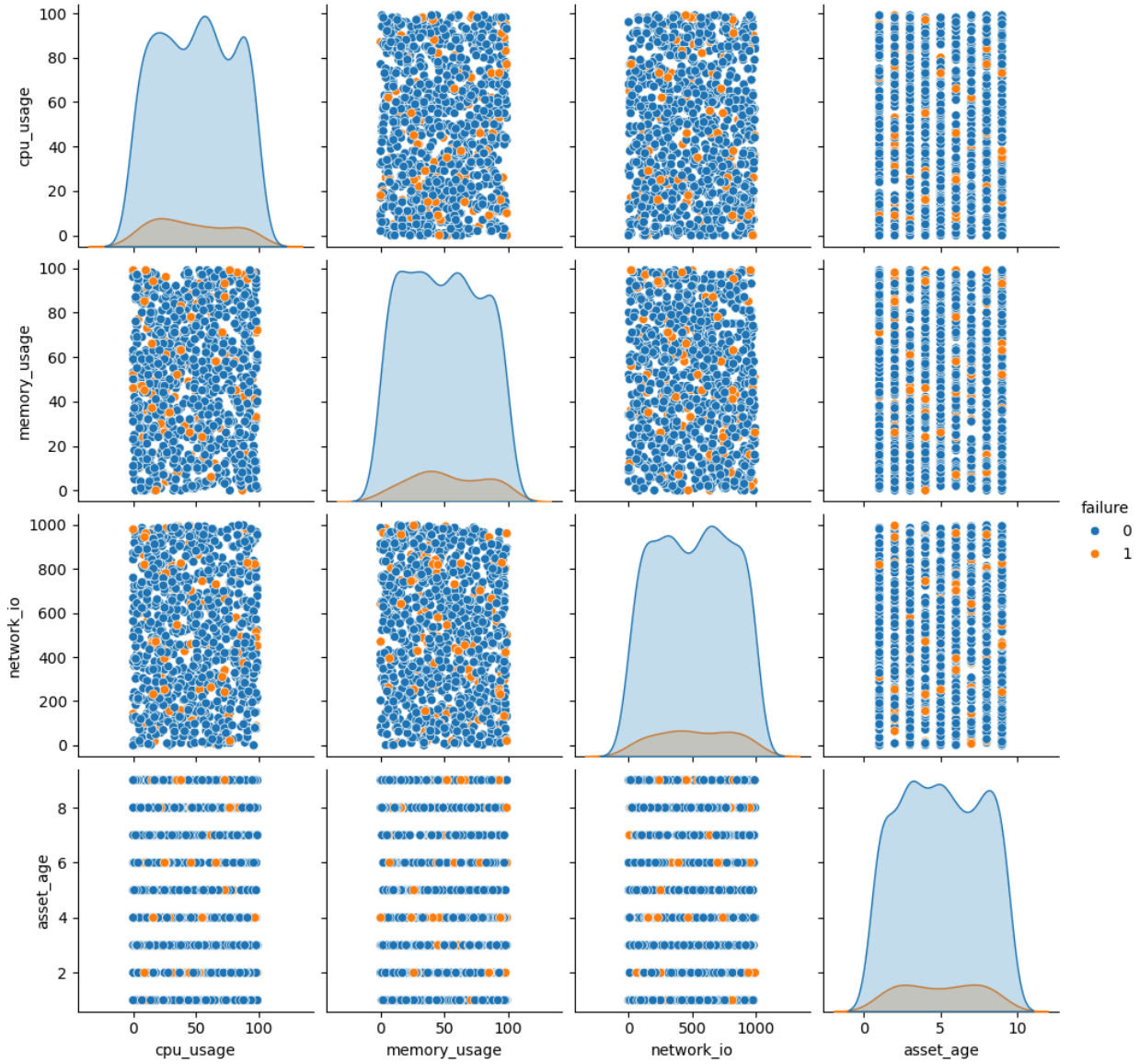
```
# Install required packages
!pip install pandas numpy scikit-learn matplotlib seaborn psutil

# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import psutil
import time
import matplotlib.pyplot as plt
import seaborn as sns

# Generate synthetic dataset
np.random.seed(42)
n_samples = 1000
data = {
    'cpu_usage': np.random.randint(0, 100, n_samples),
    'memory_usage': np.random.randint(0, 100, n_samples),
    'network_io': np.random.randint(0, 1000, n_samples),
    'asset_age': np.random.randint(1, 10, n_samples),
    'failure': np.random.choice([0, 1], n_samples, p=[0.9, 0.1])  # 10% failure rate
}

df = pd.DataFrame(data)

# Display the first few rows
print("First few rows of the dataset:")
print(df.head())

# Visualize the dataset
sns.pairplot(df, hue='failure')
plt.show()

# Prepare data for modeling
X = df.drop(columns=['failure'])
y = df['failure']
```

```
# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a RandomForest model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
print("\nModel Evaluation:")
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

# Asset Management: Function to get CPU and memory usage
def get_system_usage():
    cpu_usage = psutil.cpu_percent(interval=1)
    memory_info = psutil.virtual_memory()
    memory_usage = memory_info.percent
    return cpu_usage, memory_usage

# Monitor asset usage
cpu, memory = get_system_usage()
print(f"\nAsset Management:\nCPU Usage: {cpu}%\nMemory Usage: {memory}%")

# Network Monitoring: Function to get network I/O
def get_network_io():
    net_io = psutil.net_io_counters()
    return net_io.bytes_sent, net_io.bytes_recv

# Monitor network I/O
bytes_sent, bytes_recv = get_network_io()
print(f"\nNetwork Monitoring:\nBytes Sent: {bytes_sent}\nBytes Received: {bytes_recv}")

# Monitor network I/O over time
print("\nNetwork I/O over time:")
for _ in range(5):
    bytes_sent, bytes_recv = get_network_io()
    print(f"Bytes Sent: {bytes_sent}, Bytes Received: {bytes_recv}"time.sleep(1)
```

OUTPUT

First few rows of the dataset:

	cpu_usage	memory_usage	network_io	asset_age	failure
0	51	33	587	8	0
1	92	7	888	3	0
2	14	39	45	5	0
3	71	82	671	1	0
4	60	41	462	7	0



```
Asset Management:
CPU Usage: 51.0%
Memory Usage: 11.4%

Network Monitoring:
Bytes Sent: 3181301
Bytes Received: 2556195

Network I/O over time:
Bytes Sent: 3181301, Bytes Received: 2556195
Bytes Sent: 3202993, Bytes Received: 2577078
Bytes Sent: 3218982, Bytes Received: 2593875
Bytes Sent: 3234971, Bytes Received: 2609735
Bytes Sent: 3250960, Bytes Received: 2625595
```