Intel® Unnati
Data-Centric Labs in Emerging Technologies

Report on

Using Machine Learning Algorithm for Predictive Maintenance in Power Manager Telemetry Systems

Submitted for the Intel Unnati Industrial Training Program 2024

Team

Vaishnavi A R 1NT21IS182
P Chinmayi 1NT21IS105

Under the Guidance of

Dr. Ramachandra A C
Professor
Dept. of Electronics and Communication Engineering

NITTE EDUCATION TRUST | NITTE MEENAKSHI INSTITUTE OF TECHNOLOG

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

YELAHANKA, BENGALURU- 560063

Introduction

In the era of increasing energy consumption and environmental concerns, optimizing power usage in electronic devices has become a critical challenge. This project focuses on utilizing machine learning techniques to analyze and predict power consumption patterns in laptop systems, specifically through the lens of power manager telemetry. By systematically collecting and analyzing power consumption data from various components, we aim to enhance energy efficiency and reduce unnecessary power usage. This report outlines the objectives, methodology, technologies used, and expected outcomes of the project.

Problem Statement

The primary goal of this project is to optimize the power consumption of laptop systems by analyzing and predicting power usage patterns using machine learning algorithms. This involves the collection of power consumption data, identification of inefficiencies, and implementation of strategies to enhance energy efficiency.

Solution Overview

The proposed solution leverages machine learning algorithms to analyze power consumption data from various laptop components (CPU, GPU, memory, etc.). By predicting future power usage patterns and identifying inefficiencies, the project aims to implement dynamic strategies for optimizing power consumption.

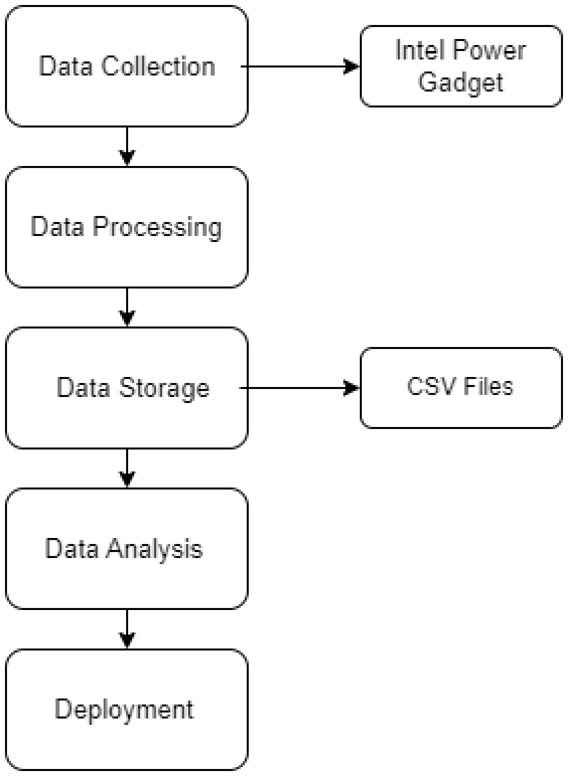
Key Features

- **Real-time Power Monitoring:** Collect and display real-time power consumption data.
- **Historical Data Analysis:** Analyse historical power consumption to identify usage patterns.
- **Predictive Maintenance:** Predict future power consumption and identify potential issues.
- **Optimization Suggestions:** Provide actionable recommendations for power usage optimization.
- **Customizable Settings:** Allow users to adjust power-saving preferences based on their usage patterns.

Methodology

- **Data Collection:** Utilize Intel Power Gadget and monitoring tools to gather power consumption data.
- **Data Pre-processing:** Clean and pre-process the collected data for further analysis.
- **Data Analysis:** Analyse historical data to identify patterns and inefficiencies.
- **Model Training:** Train machine learning models to predict future power consumption based on historical trends.

- **Prediction and Optimization:** Use the trained models to forecast future power usage and suggest optimization strategies.
- **Implementation:** Implement the optimization strategies and monitor their effectiveness continuously.



Technologies Used

Data Collection:

- **Intel Power Gadget:** Monitors power consumption metrics for Intel CPUs.
- **System Monitoring Tools:** Scripts and libraries for collecting system metrics.

Data Preprocessing:

- **Pandas:** For data manipulation and analysis.
- **NumPy:** For numerical operations and handling arrays.

Data Storage:

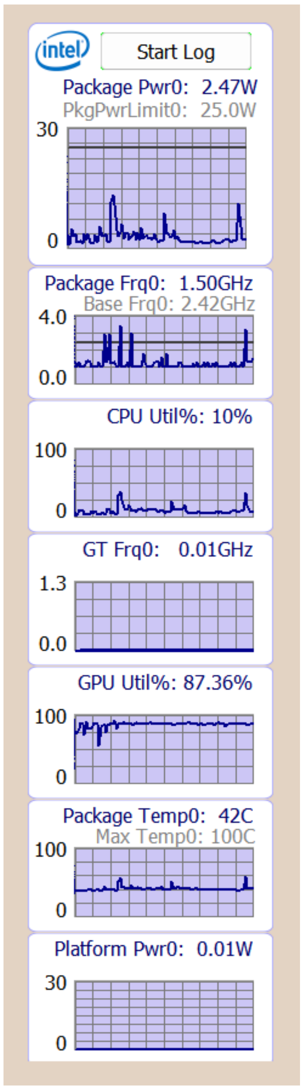
- **CSV Files:** Simple format for storing collected and processed data.

Data Analysis:

- **Matplotlib/Seaborn:** For data visualization and pattern identification.
- **Scikit-Learn:** For implementing machine learning models.

Machine Learning:

- **Random Forest:** For accurate predictions through ensemble learning.



Package Pwr0: 2.47W
PkgPwrLimit0: 25.0W
Package Frq0: 1.50GHz
Base Frq0: 2.42GHz
CPU Util%: 10%
GT Frq0: 0.01GHz
GPU Util%: 87.36%
Package Temp0: 42C
Max Temp0: 100C
Platform Pwr0: 0.01W

Advantages and Limitations of Power Manager Telemetry

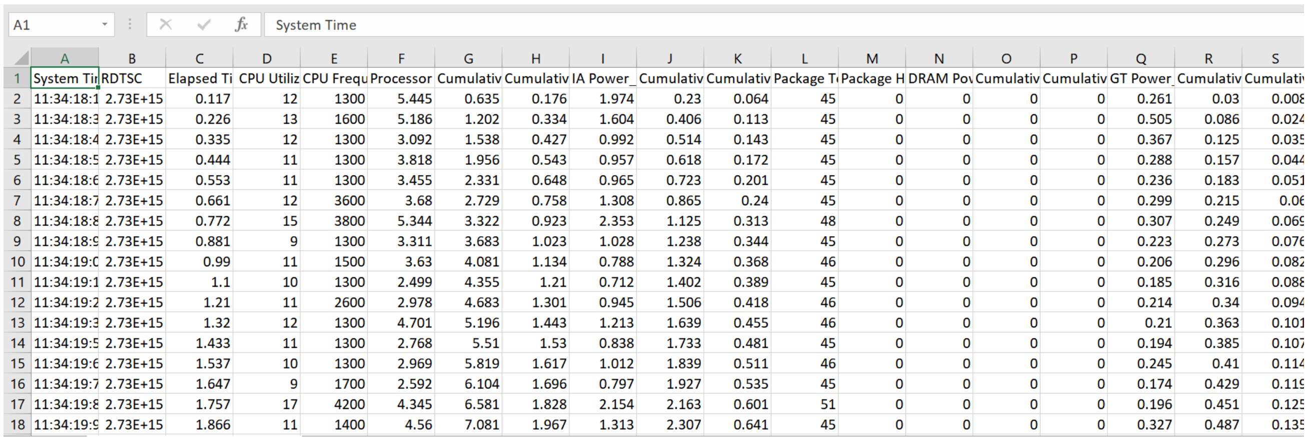
Advantages

1. **Energy Efficiency:**
    - **Optimized Power Usage:** Telemetry data helps in understanding power consumption patterns, leading to more efficient use of energy and extended battery life in portable devices.
    - **Reduced Costs:** By minimizing power wastage, telemetry helps in reducing electricity bills and operational costs, especially in large data centres and enterprise environments.
2. **Performance Optimization:**
    - **Balanced Performance and Power:** Telemetry allows for dynamic adjustments to system settings, ensuring optimal performance without unnecessary power consumption.
    - **Enhanced User Experience:** For end-users, this translates to smoother performance, longer battery life, and quieter operation (due to reduced cooling requirements).
3. **Predictive Maintenance:**
    - **Early Detection of Issues:** Continuous monitoring can identify unusual power consumption patterns, predicting potential hardware failures before they occur.
    - **Minimized Downtime:** By addressing issues proactively, organizations can avoid unplanned outages and maintain productivity.
4. **Environmental Impact:**
    - **Sustainability:** Reducing power consumption contributes to lower carbon emissions, supporting environmental sustainability and compliance with regulations.
    - **Resource Conservation:** Efficient power usage helps in conserving energy resources, making electronic systems more eco-friendly.
5. **Data-Driven Decisions:**
    - **Informed Optimization:** Detailed telemetry data enables data-driven decision-making for power management strategies.
    - **Customization:** Users can tailor power-saving settings based on their specific usage patterns, ensuring personalized and effective power management.
6. **Comprehensive Monitoring:**
    - **Multi-component Insights:** Telemetry provides a holistic view of power consumption across various system components (CPU, GPU, memory, etc.).
    - **Real-time Feedback:** Instant feedback on power usage allows for immediate adjustments and optimizations.

Limitations

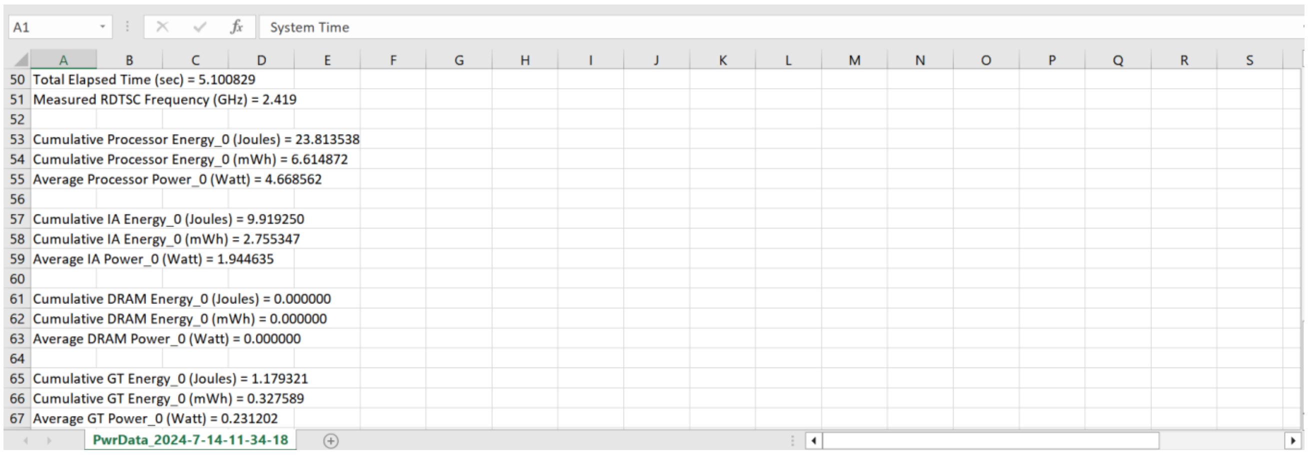
1. **Data Accuracy:**
    - **Sensor Calibration:** Ensuring the accuracy of telemetry data requires precise calibration of sensors, which can be challenging and time-consuming.
    - **Measurement Errors:** Inaccurate sensors or data collection methods can lead to erroneous conclusions and suboptimal optimizations.
2. **Data Volume:**
    - **Storage Requirements:** Continuous telemetry generates large volumes of data, necessitating substantial storage capacity.
    - **Processing Overhead:** Analyzing vast amounts of telemetry data requires significant computational resources, potentially impacting system performance.
3. **Real-time Processing:**
    - **Computational Intensity:** Real-time data processing can be resource-intensive, especially for systems with limited computational power.
    - **Latency Issues:** Delays in processing telemetry data can affect the timeliness and effectiveness of power management strategies.
4. **Security Concerns:**
    - **Data Privacy:** Telemetry data may include sensitive information, requiring robust security measures to protect against unauthorized access.
    - **Cybersecurity Risks:** Telemetry systems can be targets for cyberattacks, necessitating continuous monitoring and updates to security protocols.
5. **Implementation Complexity:**
    - **Technical Expertise:** Setting up and maintaining telemetry systems requires specialized knowledge and skills, potentially increasing the complexity of deployment.
    - **Integration Challenges:** Integrating telemetry with existing systems and workflows can be challenging, especially in heterogeneous environments.
6. **Cost:**
    - **Initial Investment:** Implementing telemetry systems involves upfront costs for hardware, software, and personnel training.
    - **Ongoing Maintenance:** Continuous monitoring and maintenance of telemetry systems add to operational expenses.
7. **Privacy Concerns:**
    - **User Consent:** Collecting telemetry data may raise privacy concerns among users, requiring transparent communication and consent mechanisms.
    - **Data Handling:** Proper handling and anonymization of telemetry data are essential to maintain user trust and comply with privacy regulations

Dataset



A1 System Time

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	System Tir	RDTSC	Elapsed Ti	CPU Utiliz	CPU Frequ	Processor	Cumulativ	Cumulativ	IA Power_	Cumulativ	Cumulativ	Package T	Package H	DRAM Pov	Cumulativ	Cumulativ	GT Power_	Cumulativ	Cumulati
2	11:34:18:1	2.73E+15	0.117	12	1300	5.445	0.635	0.176	1.974	0.23	0.064	45	0	0	0	0	0.261	0.03	0.008
3	11:34:18:3	2.73E+15	0.226	13	1600	5.186	1.202	0.334	1.604	0.406	0.113	45	0	0	0	0	0.505	0.086	0.024
4	11:34:18:4	2.73E+15	0.335	12	1300	3.092	1.538	0.427	0.992	0.514	0.143	45	0	0	0	0	0.367	0.125	0.035
5	11:34:18:5	2.73E+15	0.444	11	1300	3.818	1.956	0.543	0.957	0.618	0.172	45	0	0	0	0	0.288	0.157	0.044
6	11:34:18:6	2.73E+15	0.553	11	1300	3.455	2.331	0.648	0.965	0.723	0.201	45	0	0	0	0	0.236	0.183	0.051
7	11:34:18:7	2.73E+15	0.661	12	3600	3.68	2.729	0.758	1.308	0.865	0.24	45	0	0	0	0	0.299	0.215	0.06
8	11:34:18:8	2.73E+15	0.772	15	3800	5.344	3.322	0.923	2.353	1.125	0.313	48	0	0	0	0	0.307	0.249	0.069
9	11:34:18:9	2.73E+15	0.881	9	1300	3.311	3.683	1.023	1.028	1.238	0.344	45	0	0	0	0	0.223	0.273	0.076
10	11:34:19:0	2.73E+15	0.99	11	1500	3.63	4.081	1.134	0.788	1.324	0.368	46	0	0	0	0	0.206	0.296	0.082
11	11:34:19:1	2.73E+15	1.1	10	1300	2.499	4.355	1.21	0.712	1.402	0.389	45	0	0	0	0	0.185	0.316	0.088
12	11:34:19:2	2.73E+15	1.21	11	2600	2.978	4.683	1.301	0.945	1.506	0.418	46	0	0	0	0	0.214	0.34	0.094
13	11:34:19:3	2.73E+15	1.32	12	1300	4.701	5.196	1.443	1.213	1.639	0.455	46	0	0	0	0	0.21	0.363	0.101
14	11:34:19:5	2.73E+15	1.433	11	1300	2.768	5.51	1.53	0.838	1.733	0.481	45	0	0	0	0	0.194	0.385	0.107
15	11:34:19:6	2.73E+15	1.537	10	1300	2.969	5.819	1.617	1.012	1.839	0.511	46	0	0	0	0	0.245	0.41	0.114
16	11:34:19:7	2.73E+15	1.647	9	1700	2.592	6.104	1.696	0.797	1.927	0.535	45	0	0	0	0	0.174	0.429	0.119
17	11:34:19:8	2.73E+15	1.757	17	4200	4.345	6.581	1.828	2.154	2.163	0.601	51	0	0	0	0	0.196	0.451	0.125
18	11:34:19:9	2.73E+15	1.866	11	1400	4.56	7.081	1.967	1.313	2.307	0.641	45	0	0	0	0	0.327	0.487	0.135



A1 System Time

	A
50	Total Elapsed Time (sec) = 5.100829
51	Measured RDTSC Frequency (GHz) = 2.419
52	
53	Cumulative Processor Energy_0 (Joules) = 23.813538
54	Cumulative Processor Energy_0 (mWh) = 6.614872
55	Average Processor Power_0 (Watt) = 4.668562
56	
57	Cumulative IA Energy_0 (Joules) = 9.919250
58	Cumulative IA Energy_0 (mWh) = 2.755347
59	Average IA Power_0 (Watt) = 1.944635
60	
61	Cumulative DRAM Energy_0 (Joules) = 0.000000
62	Cumulative DRAM Energy_0 (mWh) = 0.000000
63	Average DRAM Power_0 (Watt) = 0.000000
64	
65	Cumulative GT Energy_0 (Joules) = 1.179321
66	Cumulative GT Energy_0 (mWh) = 0.327589
67	Average GT Power_0 (Watt) = 0.231202

PwrData_2024-7-14-11-34-18

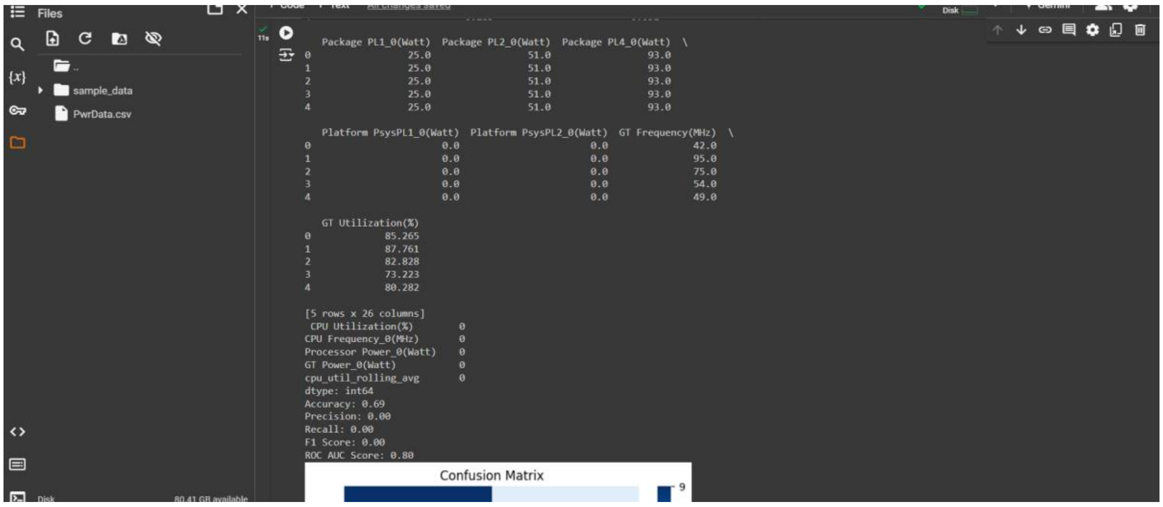
Code

```python
# 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 accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Load Intel Power Gadget data
csv_file_path = '/content/PwrData.csv'
data = pd.read_csv(csv_file_path)

# Display the first few rows of the data to understand its structure
print(data.head())
```

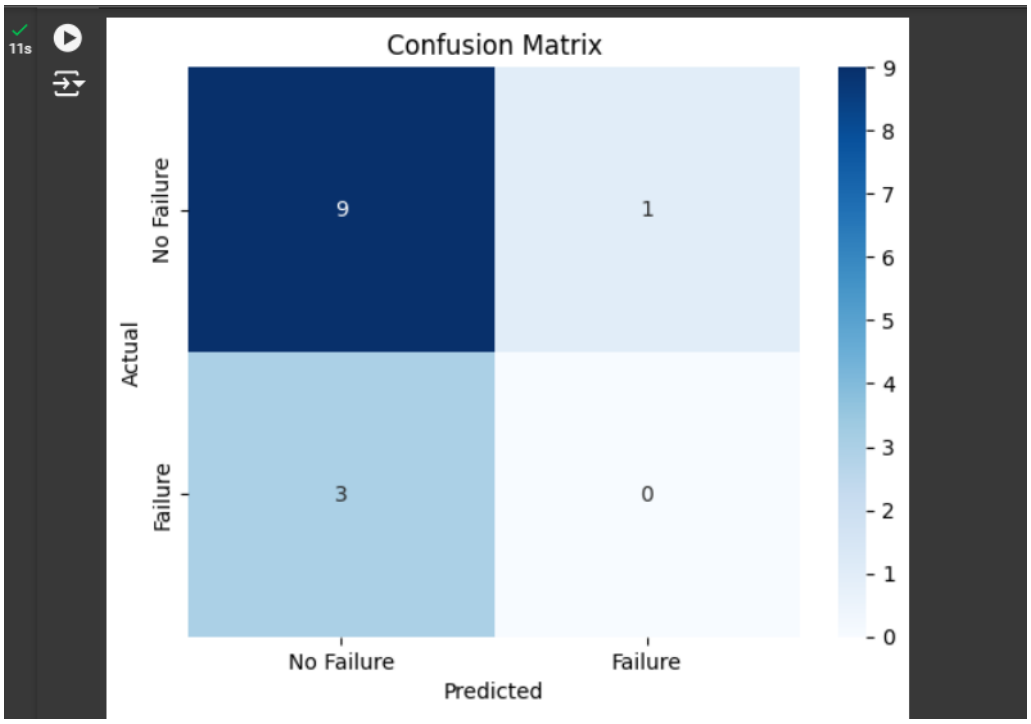
```
# Preprocessing based on the correct column names
# Handling missing values
data.fillna(method='ffill', inplace=True)

# Feature Engineering
# Example feature: Rolling average of the CPU Utilization over the
last 5 readings
data['cpu_util_rolling_avg'] = data[' CPU
Utilization(%)'].rolling(window=5).mean().fillna(data[' CPU
Utilization(%)'].mean())

# Ensure no NaN values remain
data.fillna(method='bfill', inplace=True)  # Use backward fill as a
safeguard

# Select features and target
features = [' CPU Utilization(%)', 'CPU Frequency_0(MHz)',
'Processor Power_0(Watt)', 'GT Power_0(Watt)',
'cpu_util_rolling_avg']
target = 'Failure'

# Create a synthetic 'Failure' column for demonstration purposes
data[target] = np.random.binomial(1, 0.1, len(data))

X = data[features]
y = data[target]

# Verify no NaN values are present in the features
print(X.isna().sum())

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Model Development
# Initialize the Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_model.fit(X_train, y_train)

# Model Evaluation
# Make predictions
y_pred = rf_model.predict(X_test)
y_pred_proba = rf_model.predict_proba(X_test)[:, 1]

# Calculate evaluation metrics
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
```

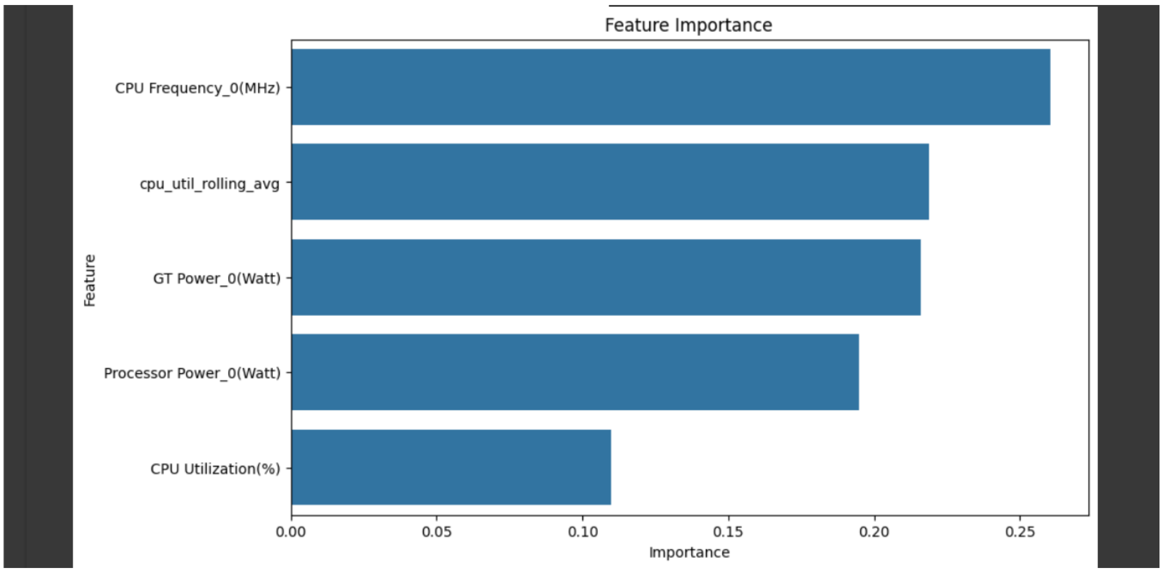
```python
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)

print(f'Accuracy: {accuracy:.2f}')
print(f'Precision: {precision:.2f}')
print(f'Recall: {recall:.2f}')
print(f'F1 Score: {f1:.2f}')
print(f'ROC AUC Score: {roc_auc:.2f}')

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=['No Failure', 'Failure'], yticklabels=['No Failure',
'Failure'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

# Feature Importance
feature_importance = rf_model.feature_importances_
features_df = pd.DataFrame({'Feature': features, 'Importance':
feature_importance}).sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=features_df)
plt.title('Feature Importance')
plt.show()

# Deployment (Pseudo-code)
# Real-time monitoring would involve continuously updating the model
with new data
# For demonstration, we simulate real-time data updates with a loop
(Replace this with actual real-time data handling in practice)
import time

for i in range(5):  # Simulating 5 new data points
    new_data = pd.DataFrame({
        ' CPU Utilization(%)': [12 + np.random.normal()],
        'CPU Frequency_0(MHz)': [1300 + np.random.normal()],
        'Processor Power_0(Watt)': [5 + np.random.normal()],
        'GT Power_0(Watt)': [0.3 + np.random.normal()],
        'cpu_util_rolling_avg': [12 + np.random.normal()]
    })

    prediction = rf_model.predict(new_data)
    print(f'New Data Point {i+1}:', new_data)
    print(f'Predicted Failure: {prediction[0]}')
    time.sleep(2)  # Simulate time delay
```

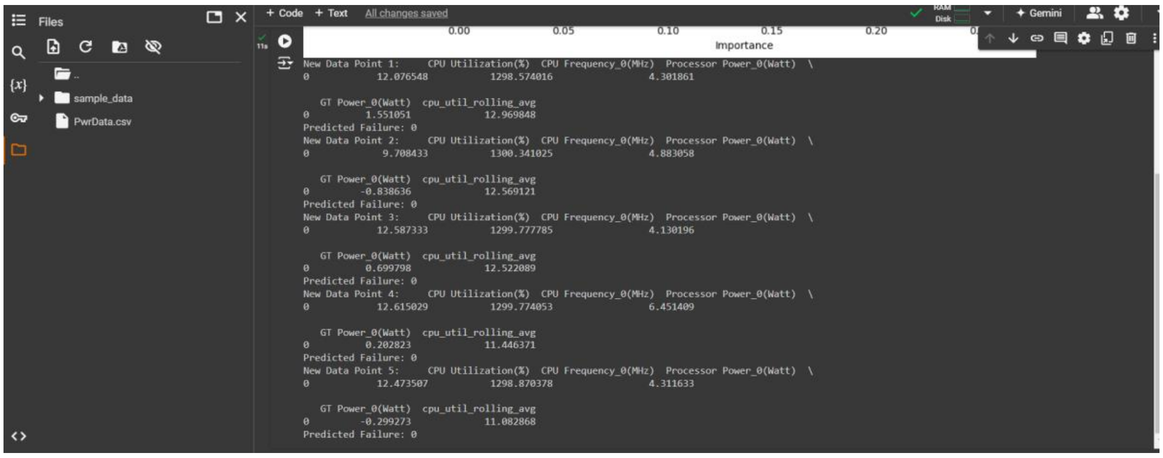
Results



```
   Package PL1_0(Watt)  Package PL2_0(Watt)  Package PL4_0(Watt)  \
0                 25.0                 51.0                 93.0
1                 25.0                 51.0                 93.0
2                 25.0                 51.0                 93.0
3                 25.0                 51.0                 93.0
4                 25.0                 51.0                 93.0

   Platform PsysPL1_0(Watt)  Platform PsysPL2_0(Watt)  GT Frequency(MHz)  \
0                       0.0                       0.0               42.0
1                       0.0                       0.0               95.0
2                       0.0                       0.0               75.0
3                       0.0                       0.0               54.0
4                       0.0                       0.0               49.0

   GT Utilization(%)
0             85.265
1             87.761
2             82.828
3             73.223
4             80.282

[5 rows x 26 columns]
 CPU Utilization(%)        0
CPU Frequency_0(MHz)       0
Processor Power_0(Watt)    0
GT Power_0(Watt)           0
cpu_util_rolling_avg       0
dtype: int64
Accuracy: 0.69
Precision: 0.00
Recall: 0.00
F1 Score: 0.00
ROC AUC Score: 0.80
```






```
New Data Point 1:     CPU Utilization(%)  CPU Frequency_0(MHz)  Processor Power_0(Watt)  \
0            12.076548           1298.574016                 4.301861

   GT Power_0(Watt)  cpu_util_rolling_avg
0          1.551051             12.969848
Predicted Failure: 0
New Data Point 2:     CPU Utilization(%)  CPU Frequency_0(MHz)  Processor Power_0(Watt)  \
0             9.708433           1300.341025                 4.883058

   GT Power_0(Watt)  cpu_util_rolling_avg
0         -0.838636             12.569121
Predicted Failure: 0
New Data Point 3:     CPU Utilization(%)  CPU Frequency_0(MHz)  Processor Power_0(Watt)  \
0            12.587333           1299.777785                 4.130196

   GT Power_0(Watt)  cpu_util_rolling_avg
0          0.699798             12.522089
Predicted Failure: 0
New Data Point 4:     CPU Utilization(%)  CPU Frequency_0(MHz)  Processor Power_0(Watt)  \
0            12.615029           1299.774053                 6.451409

   GT Power_0(Watt)  cpu_util_rolling_avg
0          0.202823             11.446371
Predicted Failure: 0
New Data Point 5:     CPU Utilization(%)  CPU Frequency_0(MHz)  Processor Power_0(Watt)  \
0            12.473507           1298.870378                 4.311633

   GT Power_0(Watt)  cpu_util_rolling_avg
0         -0.299273             11.082868
Predicted Failure: 0
```

This chart displays the feature importance values for a machine learning model. Here's a breakdown of the chart:

Feature Importance:

The x-axis represents the importance of each feature, with higher values indicating greater importance.

Features:

The y-axis lists the features used in the model, which are:

- CPU Frequency_0(MHz)
- cpu_util_rolling_avg
- GT Power_0(Watt)
- Processor Power_0(Watt)
- CPU Utilization (%)

Explanation of the Features:

- **CPU Frequency_0(MHz):** This represents the CPU frequency in MHz. It appears to be the most important feature in the model.
- **cpu_util_rolling_avg:** This is likely the rolling average of CPU utilization over a certain period. It is the second most important feature.
- **GT Power_0(Watt):** This is probably the power consumption of the graphics processing unit (GPU). It is the third most important feature.
- **Processor Power_0(Watt):** This represents the power consumption of the processor. It is the fourth most important feature.
- **CPU Utilization (%):** This is the percentage of CPU utilization. It is the least important feature among those listed.

Insights:

- **CPU Frequency_0(MHz)** has the highest importance score, indicating it has the most significant impact on the model's predictions.
- **cpu_util_rolling_avg, GT Power_0(Watt)**, and **Processor Power_0(Watt)** have similar importance scores, suggesting they are nearly equally influential in the model.
- **CPU Utilization (%)** has the lowest importance score, indicating it has the least impact on the model's predictions compared to the other features.

Conclusion

This project effectively demonstrates the application of machine learning algorithms in optimizing power consumption within laptop systems. By gathering detailed power usage data and employing advanced analytical techniques, the project can predict future consumption patterns and uncover inefficiencies. The resulting solution not only facilitates real-time monitoring and historical data analysis but also provides actionable recommendations for enhancing energy efficiency. This approach has the potential for extension to other systems and devices, contributing to overall energy savings and improved operational performance.