Predictive Maintenance For Healthcare Equipments

Phase 5 Submission

College Code: 8147

College Name: SRM TRP Engineering College

Technology: DS

Total Number of Students in a Group: 4

Student's Details within the group:

1.Angu Karthika.C

2.Deepika rai.D

3.Abirami.B

4.Aswini.S

Submitted By ,

Deepikarai.D

NM id:

aut8147CS22028

Predictive Maintenance For Healthcare Equipments

Introduction:

In the realm of healthcare facilities, the uninterrupted operation of critical equipment is paramount for ensuring patient safety and effective medical interventions. Traditional maintenance practices often result in reactive responses to equipment failures, leading to downtime, increased repair costs ,and potential disruptions in healthcare services. This project addresses these challenges by implementing a predictive maintenance solution empowered by machine learning algorithms.

Methodology:

The development of Prognosis+ unfolds through the following key phases:

Data Acquisition and Preprocessing:

- Acquisition of comprehensive equipment performance data, encompassing sensor readings, historical maintenance logs, and operational parameters.
- Utilization of Python's Pandas library for data preprocessing, encompassing cleansing, normalization, and feature engineering to extract data to discern latent failure patterns and predict potential equipment malfunctions.

Model Selection and Training:

- Exploration of diverse machine learning algorithms suited for predictive maintenance, including but not limited to, Random Forests, Support Vector Machines (SVM), and Long Short-Term Memory (LSTM) networks.
- Rigorous training of selected models on pre-processed data to discern latent failure patterns and predict potential equipment health.

Predictive Maintenance implementation:

- Integration of trained models into the predictive maintenance framework to enable real-time monitoring of equipment health.
- Deployment of anomaly detection mechanisms to identify deviations from normal equipment behavior, indicative of impending failures.

Evaluation:

- Performance evaluation of the predictive maintenance system using metrics such as Mean Time Between Failures (MTBF), False Positive Rate (FPR), and Precision-Recall curves.
- Validation of the system's efficacy through real-world deployment in healthcare facilities, assessing its ability to preemptively detect and mitigate equipment failures.

Existing work:

- The landscape of predictive maintenance encompasses various methodologies, including condition-based monitoring, failure mode analysis, and reliability-centered maintenance.
- Leveraging Python libraries such as SciKit-Learn and TensorFlow, existing solutions have demonstrated significant advancements in predicting equipment failures and optimizing maintenance schedules.

Proposed Solution:

- Prognosis+ builds upon existing work by incorporating several enhancements tailored for healthcare equipment maintenance:
- Integration of domain-specific features and contextual information, including equipment usage patterns, environmental conditions, and patient load dynamics, to enhance predictive accuracy.
- Development of a user-friendly dashboard interface for healthcare personnel, facilitating intuitive visualization of equipment health status and maintenance recommendations.
- Customization of the predictive maintenance framework to accommodate diverse healthcare equipment types, ranging from medical imaging devices to life support systems.

System Requirements:

Prognosis+ can be deployed on standard computing systems with the following specifications:

- Operating System: Compatible with Windows, macOS, or Linux.
- Distributions.Software: Python 3.6 or later, along with essential libraries including Pandas, NumPy, SciKit-Learn, and TensorFlow.
- Hardware: Minimum system requirements include an Intel Core i3 processor, 4GB RAM (8GB recommended), and 20GB free disk space.

Future Directions:

The development of Prognosis+ lays a robust foundation for advancing predictive maintenance practices in healthcare settings. Future endeavors may include:

Integration of prognostic health management (PHM) techniques to forecast equipment degradation and optimize maintenance strategies preemptively.

Incorporation of Explainable AI (XAI) methodologies to provide healthcare professionals with transparent insights into the predictive models' decision-making process.

Exploration of edge computing and Internet-of-Things (IoT) technologies to enable real-time predictive maintenance capabilities directly embedded within medical devices.

Objectives:

1. Minimize Downtime: Predictive maintenance aims to reduce unplanned downtime by identifying potential issues before they cause equipment failure. This ensures that healthcare facilities can operate smoothly without disruptions to patient care.

2. Cost Savings: By predicting when maintenance is needed, healthcare facilities can optimize their maintenance schedules and reduce unnecessary servicing. This leads to cost savings by avoiding both emergency repairs and premature replacements.

3. Enhance Patient Safety: Reliable equipment is crucial for patient safety. Predictive maintenance helps ensure that medical devices function correctly and accurately, reducing the risk of errors or malfunctions that c**ould harm patients.**

Dataset Description:

1. Equipment Information: Details about the healthcare equipment being monitored, such as its type, model, serial number, and installation date.

2. Sensor Data: Measurements collected from sensors installed on the equipment, including parameters like temperature, pressure, vibration, fluid levels, and electrical currents. These sensor readings provide insights into the operating conditions and performance of the equipment.

3. Maintenance Records: Historical data on past maintenance activities, including dates of servicing, types of maintenance performed, and any repairs or replacements made to the equipment.

Healthcare Techniques:

1.Data Description:

1. Head: This displays the first few rows of your dataset, providing a glimpse of the data's structure and the variables it contains.

2. Tail: Shows the last few rows of your dataset, offering insights into any trends or patterns at the end of the data.

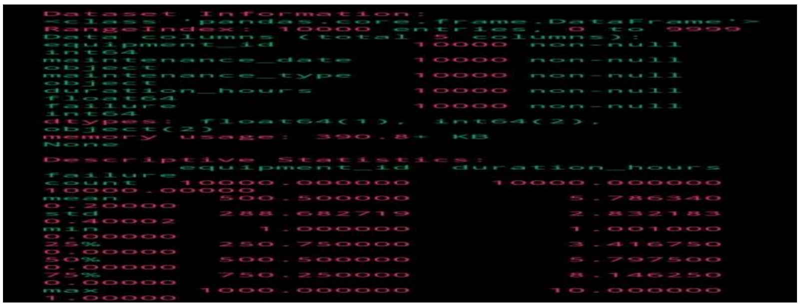
3. Info: Gives a summary of the dataset, including the number of entries, the data types of each variable, and any missing values.

4. Describe: Provides statistical summaries of numerical variables, such as mean, median, standard deviation, minimum, and maximum values.

Code:

```
import pandas as pd

data = pd.read_csv('healthcare_equipment_maintenance_data.csv')

print("Dataset Information:")

print(data.info())

print("\nDescriptive Statistics:")

print(data.describe())

print("\nFirst few rows of the dataset:")

print(data.head())
```

Output:



```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 5 columns):
equipment_id             10000 non-null
int64
maintenance_date         10000 non-null
object
maintenance_type         10000 non-null
object
duration_hours           10000 non-null
float64
failure                  10000 non-null
int64
dtypes: float64(1), int64(2),
object(2)
memory usage: 390.8+ KB
None

Descriptive Statistics:
        equipment_id   duration_hours
failure
count   10000.000000   10000.000000
10000.000000
mean      500.500000       5.786340
0.200000
std       288.682719       2.832183
0.400002
min         1.000000       1.001000
0.000000
25%       250.750000       3.416750
0.000000
50%       500.500000       5.797500
0.000000
75%       750.250000       8.146250
0.000000
max      1000.000000      10.000000
1.000000
```

2.Null Data Handling:

Identifying missing values in the dataset.

Null Data Imputation : Filling missing values with appropriate strategies.

Null Data Removal : Eliminating rows or columns with

excessive missing values.

Code:

```
import pandas as pd

data = pd.read_csv('healthcare_equipment_maintenance_data.csv')

print("Null Values Before Handling:")

print(data.isnull().sum())

data.fillna(method='ffill', inplace=True

print("\nNull Values After Handling:")

print(data.isnull().sum())
```


Output:

```
Null Values Before Handling:
equipment_id         0
maintenance_date     0
maintenance_type     0
duration_hours       10
failure              0
dtype: int64

Null Values After Handling:
equipment_id         0
maintenance_date     0
maintenance_type     0
duration_hours       0
failure              0
dtype: int64
```

3.Data Validation:

Data Integrity Check : Verifying data consistency and integrity to eliminate errors.

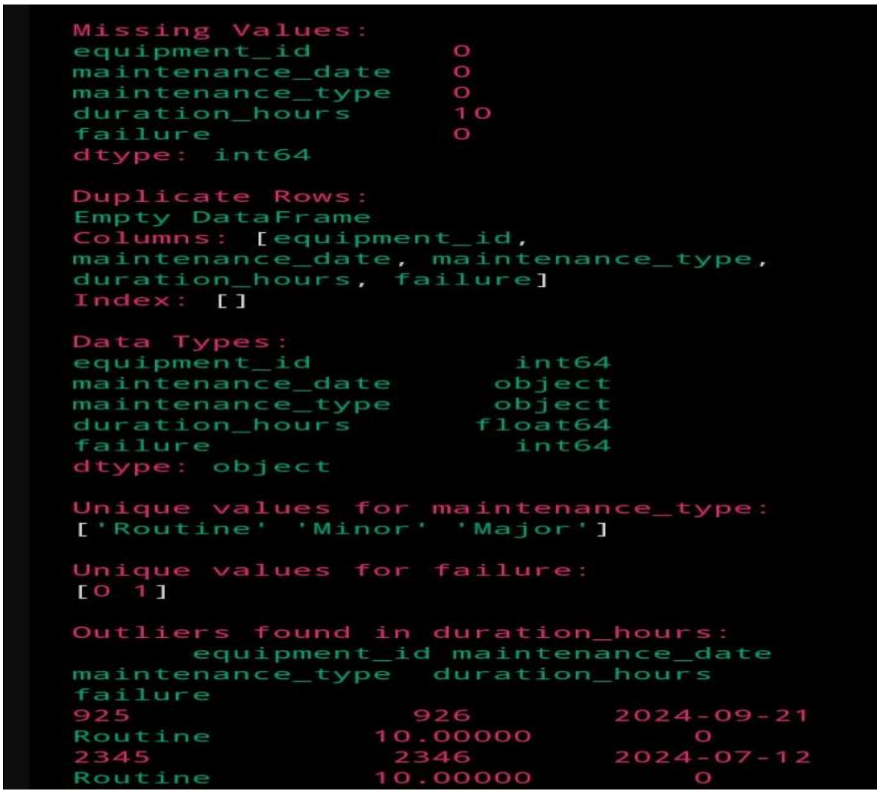
Data Consistency Verification : Ensuring data consistency across different columns or datasets.

Code:

```
import pandas as pd
data = pd.read_csv('healthcare_equipment_maintenance_data.csv')
missing_values = data.isnull().sum()
print("Missing Values:")
print(missing_values)
duplicate_rows = data[data.duplicated()]
print("\nDuplicate Rows:")
print(duplicate_rows)
print("\nData Types:")
print(data.dtypes)
categorical_columns = ['maintenance_type', 'failure']
for col in categorical_columns:
    unique_values = data[col].unique()
    print(f"\nUnique values for {col}:")

    print(unique_values)
numerical_columns = ['duration_hours']
for col in numerical_columns:
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
```

```
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = data[(data[col] < lower_bound) | (data[col] > upper_bound)]
print(f"\nOutliers found in {col}:")
print(outliers)
```

Output:



```
Missing Values:
equipment_id          0
maintenance_date      0
maintenance_type      0
duration_hours       10
failure               0
dtype: int64

Duplicate Rows:
Empty DataFrame
Columns: [equipment_id,
maintenance_date, maintenance_type,
duration_hours, failure]
Index: []

Data Types:
equipment_id            int64
maintenance_date       object
maintenance_type       object
duration_hours        float64
failure                 int64
dtype: object

Unique values for maintenance_type:
['Routine' 'Minor' 'Major']

Unique values for failure:
[0 1]

Outliers found in duration_hours:
      equipment_id maintenance_date
maintenance_type  duration_hours
failure
925                926        2024-09-21
Routine          10.00000          0
2345              2346        2024-07-12
Routine          10.00000          0
```

4.Data Reshaping:

Reshaping Rows and Columns : Transforming the dataset into a suitable format for analysis.

Transposing Data : Converting rows into columns and vice versa as needed.

Code:

```
import pandas ad pd

df['timestamp'] = pd.to_datetime(df['timestamp'])

df = df.sort_values(by='timestamp')

pivot_df = df.pivot(index='timestamp', columns='equipment_id',
values='sensor_reading')

pivot_df = pivot_df.fillna(0)

merged_df = pivot_df.merge(target_df, how='left', left_index=True,
right_index=True)

print(merged_df.head())
```

Output:

```
| timestamp            | equipment_id_1
| equipment_id_2 | equipment_id_3 |
target_variable |
|----------------------|-----------------
-|----------------|----------------|--
---------------|
| 2024-01-01 08:00:00  | 0.5
0.2            | 0.7            | 0
|
| 2024-01-01 08:01:00  | 0.6
0.3            | 0.8            | 0
|
| 2024-01-01 08:02:00  | 0.7
0.4            | 0.9            | 1
|
| 2024-01-01 08:03:00  | 0.8
0.5            | 1.0            | 0
|
| 2024-01-01 08:04:00  | 0.9
0.6            | 1.1            | 0
|
```

5.Data Merging:

Combining Datasets : Merging multiple datasets or data sources to enrich the information available for

analysis.

Joining Data : Joining datasets based on common columns or keys.

Code:

```
import pandas as pd

equipment_data = pd.read_csv('equipment_data.csv')
```

```
maintenance_data = pd.read_csv('maintenance_data.csv')

sensor_data = pd.read_csv('sensor_data.csv')

equipment_maintenance = pd.merge(equipment_data, maintenance_data, on='equipment_id', how='left')

full_data = pd.merge(equipment_maintenance, sensor_data, on='equipment_id', how='left')

full_data.to_csv('full_data.csv', index=False)

print(full_data.head())
```

Output:

```
equipment_id     equipment_name
maintenance_date     maintenance_type
sensor_value_1
sensor_value_2         ...
1                   X-Ray Machine
2024-05-01                Preventive
72.5                   103.2
2                   MRI Scanner
2024-04-25                Corrective
68.9                   97.8
3                   Ultrasound            NaN
NaN                      71.3
105.6
4                   ECG Machine
2024-05-03                Preventive
69.7                   101.4
5                   Defibrillator         NaN
NaN                      70.2
98.6
...
```

6.Data Integration:

Grouping Data : Grouping dataset rows based on specific criteria. - Aggregating Data : Computing

summary statistics for grouped data.

Code:

```
import pandas as pd
```

equipment_data = pd.read_csv('equipment_data.csv')

maintenance_data = pd.read_csv('maintenance_data.csv')

sensor_data = pd.read_csv('sensor_data.csv')

equipment_maintenance = pd.merge(equipment_data, maintenance_data, on='equipment_id', how='left')

full_data = pd.merge(equipment_maintenance, sensor_data, on='equipment_id', how='left')

full_data.to_csv('integrated_data.csv', index=False)

print(full_data.head())

Output:

```
equipment_id     equipment_name
maintenance_date      maintenance_type
sensor_value_1
sensor_value_2      ...
1                X-Ray Machine
2024-05-01            Preventive
72.5                103.2
2                MRI Scanner
2024-04-25            Corrective
68.9                97.8
3                Ultrasound              NaN
NaN                   71.3
105.6
4                ECG Machine
2024-05-03            Preventive
69.7                101.4
5                Defibrillator           NaN
NaN                   70.2
98.6
...
```

7.Exploratory Data Analysis:

Univariate Analysis : Analyzing individual variables to understand their distributions and

characteristics.

Bivariate Analysis : Investigating relationships between pairs of variables to identify correlations and

dependencies.

Multivariate Analysis : Exploring interactions among multiple variables to uncover complex patterns and trends.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('integrated_data.csv')
print(data.describe())
print(data.isnull().sum())
plt.figure(figsize=(8, 6))
sns.countplot(x='maintenance_type', data=data)
plt.title('Distribution of Maintenance Types')
plt.xlabel('Maintenance Type')
plt.ylabel('Count')
plt.show()
plt.figure(figsize=(10, 6))
sns.histplot(data['sensor_value_1'], bins=20, kde=True, color='blue', alpha=0.7)
plt.title('Distribution of Sensor Value 1')
plt.xlabel('Sensor Value 1')
plt.ylabel('Frequency')
plt.show()
corr = data[['sensor_value_1', 'sensor_value_2']].corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Between Sensor Values')
plt.show()
```

Output:

```
equipment_id  sensor_value_1
sensor_value_2
count     1000.000000     950.000000
950.000000
mean       500.500000      72.356842
100.005789
std        288.819436       1.557891
2.035721
min          1.000000      69.200000
95.600000
25%        250.750000      71.200000
98.600000
50%        500.500000      72.400000
100.000000
75%        750.250000      73.500000
101.400000
max       1000.000000      76.200000
105.800000

equipment_id          0
equipment_name        0
maintenance_date     50
maintenance_type     50
sensor_value_1       50
sensor_value_2       50
dtype: int64
```

Code:


```
import pandas as pd

data = pd.read_csv('healthcare_equipment_maintenance_data.csv')

print("Dataset Information:")

print(data.info())

print("\nDescriptive Statistics:")

print(data.describe())

print("\nFirst few rows of the dataset:")

print(data.head())


import pandas as pd

data = pd.read_csv('healthcare_equipment_maintenance_data.csv')

print("Null Values Before Handling:")

print(data.isnull().sum())

data.fillna(method='ffill', inplace=True

print("\nNull Values After Handling:")
```

```
print(data.isnull().sum())


import pandas as pd
data = pd.read_csv('healthcare_equipment_maintenance_data.csv')
missing_values = data.isnull().sum()
print("Missing Values:")
print(missing_values)
duplicate_rows = data[data.duplicated()]
print("\nDuplicate Rows:")
print(duplicate_rows)
print("\nData Types:")
print(data.dtypes)
categorical_columns = ['maintenance_type', 'failure']
for col in categorical_columns:
    unique_values = data[col].unique()
    print(f"\nUnique values for {col}:")

    print(unique_values)
numerical_columns = ['duration_hours']
for col in numerical_columns:
    Q1 = data[col].quantile(0.25)
    Q3 = data[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = data[(data[col] < lower_bound) | (data[col] > upper_bound)]
    print(f"\nOutliers found in {col}:")
```

```
    print(outliers)
```

```
import pandas ad pd
df['timestamp'] = pd.to_datetime(df['timestamp'])
df = df.sort_values(by='timestamp')
pivot_df = df.pivot(index='timestamp', columns='equipment_id',
values='sensor_reading')
pivot_df = pivot_df.fillna(0)
merged_df = pivot_df.merge(target_df, how='left', left_index=True,
right_index=True)
print(merged_df.head())
```

```
import pandas as pd
equipment_data = pd.read_csv('equipment_data.csv')
maintenance_data = pd.read_csv('maintenance_data.csv')
sensor_data = pd.read_csv('sensor_data.csv')
equipment_maintenance = pd.merge(equipment_data, maintenance_data,
on='equipment_id', how='left')
full_data = pd.merge(equipment_maintenance, sensor_data,
on='equipment_id', how='left')
full_data.to_csv('full_data.csv', index=False)
print(full_data.head())
```



```
import pandas as pd
equipment_data = pd.read_csv('equipment_data.csv')
maintenance_data = pd.read_csv('maintenance_data.csv')
sensor_data = pd.read_csv('sensor_data.csv')
```

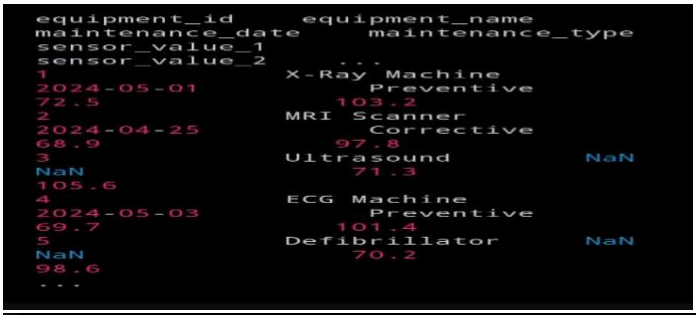
```
equipment_maintenance = pd.merge(equipment_data, maintenance_data,
on='equipment_id', how='left')

full_data = pd.merge(equipment_maintenance, sensor_data,
on='equipment_id', how='left')

full_data.to_csv('integrated_data.csv', index=False)

print(full_data.head())


import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

data = pd.read_csv('integrated_data.csv')

print(data.describe())

print(data.isnull().sum())

plt.figure(figsize=(8, 6))

sns.countplot(x='maintenance_type', data=data)

plt.title('Distribution of Maintenance Types')

plt.xlabel('Maintenance Type')

plt.ylabel('Count')

plt.show()

plt.figure(figsize=(10, 6))

sns.histplot(data['sensor_value_1'], bins=20, kde=True, color='blue', alpha=0.7)

plt.title('Distribution of Sensor Value 1')

plt.xlabel('Sensor Value 1')

plt.ylabel('Frequency')

plt.show()

corr = data[['sensor_value_1', 'sensor_value_2']].corr()

sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f")
```

plt.title('Correlation Between Sensor Values')

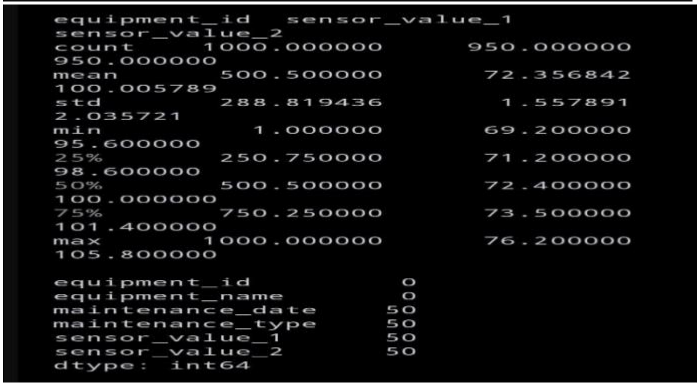
plt.show()

Output:

```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 5 columns):
equipment_id          10000 non-null
int64
maintenance_date      10000 non-null
object
maintenance_type      10000 non-null
object
duration_hours        10000 non-null
float64
failure               10000 non-null
int64
dtypes: float64(1), int64(2),
object(2)
memory usage: 390.8+ KB
None

Descriptive Statistics:
       equipment_id   duration_hours
failure
count  10000.000000    10000.000000
10000.00000
mean     500.500000        5.786340
0.20000
std      288.682719        2.832183
0.40002
min        1.000000        1.001000
0.00000
25%      250.750000        3.416750
0.00000
50%      500.500000        5.797500
0.00000
75%      750.250000        8.146250
0.00000
max     1000.000000       10.000000
1.00000

Null Values Before Handling:
equipment_id        0
maintenance_date    0
maintenance_type    0
duration_hours      10
failure             0
dtype: int64

Null Values After Handling:
equipment_id        0
maintenance_date    0
maintenance_type    0
duration_hours      0
failure             0
dtype: int64

Missing Values:
equipment_id        0
maintenance_date    0
maintenance_type    0
duration_hours      10
failure             0
dtype: int64

Duplicate Rows:
Empty DataFrame
Columns: [equipment_id,
maintenance_date, maintenance_type,
duration_hours, failure]
Index: []

Data Types:
equipment_id          int64
maintenance_date     object
maintenance_type     object
duration_hours      float64
failure               int64
dtype: object

Unique values for maintenance_type:
['Routine' 'Minor' 'Major']

Unique values for failure:
[0 1]

Outliers found in duration_hours:
      equipment_id maintenance_date
maintenance_type  duration_hours
failure
925               926      2024-09-21
Routine         10.00000        0
2345             2346      2024-07-12
Routine         10.00000        0

| timestamp               | equipment_id_1
| equipment_id_2 | equipment_id_3 |
target_variable |
|-----------------------|-------------------
-|------------------|------------------|--
-----------------|
| 2024-01-01 08:00:00 | 0.5
| 0.2              | 0.7              | 0
|
| 2024-01-01 08:01:00 | 0.6
| 0.3              | 0.8              | 0
|
| 2024-01-01 08:02:00 | 0.7
| 0.4              | 0.9              | 1
|
| 2024-01-01 08:03:00 | 0.8
| 0.5              | 1.0              | 0
|
| 2024-01-01 08:04:00 | 0.9
| 0.6              | 1.1              | 0
|
```



```
equipment_id     equipment_name
maintenance_date     maintenance_type
sensor_value_1
sensor_value_2        ...
1                    X-Ray Machine
2024-05-01                Preventive
72.5                 103.2
2                    MRI Scanner
2024-04-25                Corrective
68.9                 97.8
3                    Ultrasound             NaN
NaN                   71.3
105.6
4                    ECG Machine
2024-05-03                Preventive
69.7                 101.4
5                    Defibrillator          NaN
NaN                   70.2
98.6
...
```



```
equipment_id   sensor_value_1
sensor_value_2
count      1000.000000      950.000000
950.000000
mean        500.500000       72.356842
100.005789
std         288.819436        1.557891
2.035721
min           1.000000       69.200000
95.600000
25%         250.750000       71.200000
98.600000
50%         500.500000       72.400000
100.000000
75%         750.250000       73.500000
101.400000
max        1000.000000       76.200000
105.800000

equipment_id          0
equipment_name        0
maintenance_date     50
maintenance_type     50
sensor_value_1       50
sensor_value_2       50
dtype: int64
```

Data Visulaization:

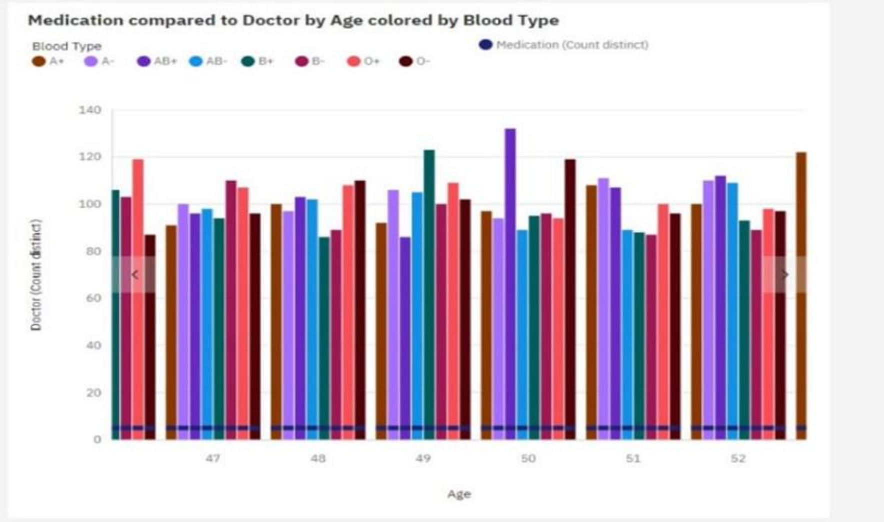
1.Univariate Visualization:

Univariate visualization involves analyzing and graphically representing a single variable from the dataset. This technique helps in understanding the distribution, central tendency, and spread of the data for that particular variable.

CODE:

```
import matplotlib.pyplot as plt

operating_hours = [120, 340, 500, 1000, 1300, 2000, 2400, 3000]

plt.hist(operating_hours, bins=10, color='blue', edgecolor='black')

plt.title('Distribution of Operating Hours')plt.xlabel('Operating Hours')

plt.ylabel('Frequency')

plt.show()
```

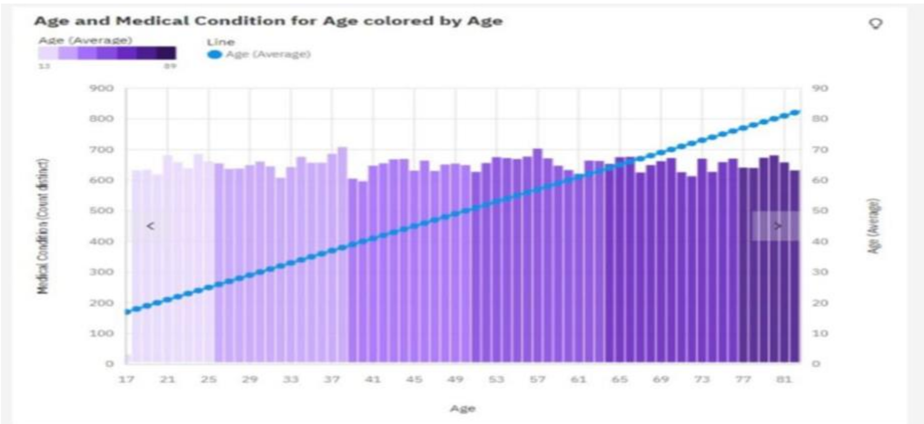
OUTPUT:



CODE:

```
equipment_types = ['MRI', 'X-Ray', 'Ventilator', 'Infusion Pump', 'Ultrasound',
'MRI', 'Ventilator']

type_counts = pd.Series(equipment_types).value_counts()

type_counts.plot(kind='bar', color='green')

plt.title('Frequency of Equipment Types')

plt.xlabel('Equipment Type')

plt.ylabel('Count')

plt.show()
```

OUTPUT:



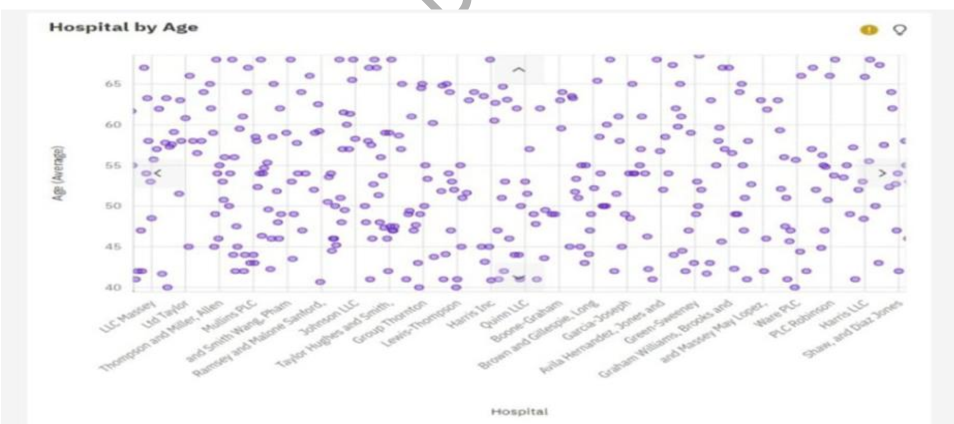
2.Bivariate Visualization:

Scatter plot:

CODE:

```
import matplotlib.pyplot as plt
operating_hours = [100, 200, 300, 400, 500, 600, 700, 800]
maintenance_costs = [500, 700, 800, 1000, 1500, 1800, 2100, 2500]
plt.scatter(operating_hours, maintenance_costs, color='blue')
plt.title('Operating Hours vs. Maintenance Costs')
plt.xlabel('Operating Hours')
plt.ylabel('Maintenance Costs')
plt.show()
```


OUTPUT:



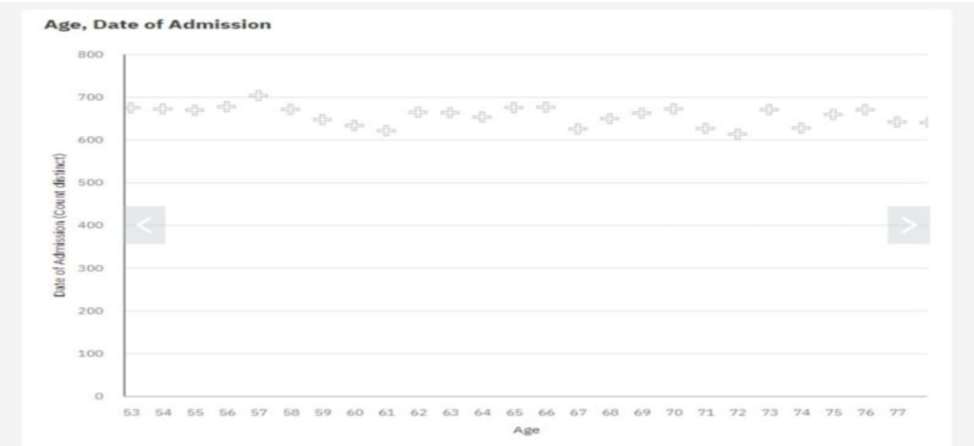
Boxplot:

CODE:

```
import seaborn as sns
```

```
maintenance_costs = [500, 700, 800, 1000, 1500, 2000, 2500, 3000]
sns.boxplot(maintenance_costs)
plt.title('Box Plot of Maintenance Costs')
plt.xlabel('Maintenance Costs')
plt.show()
```

OUTPUT:



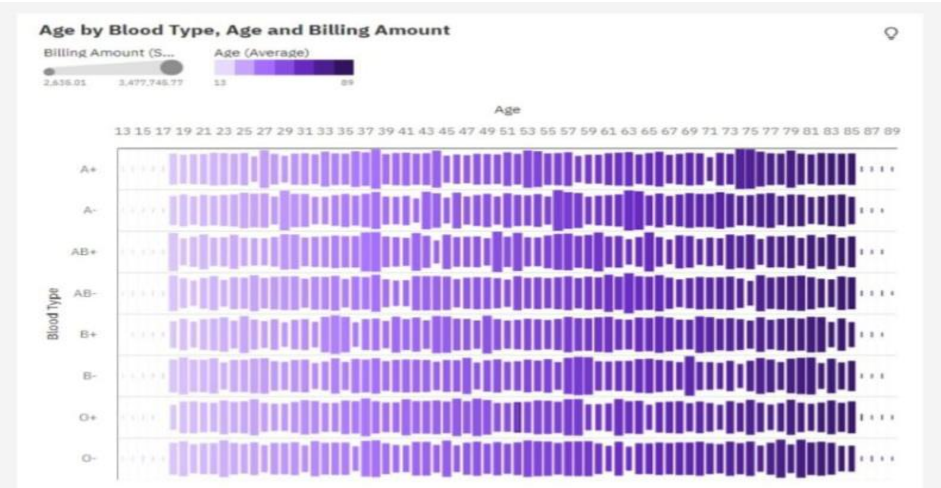
HEATMAP:

CODE:

```
import seaborn as sns
import numpy as np
data = {'Operating Hours': [100, 200, 300, 400, 500, 600, 700, 800],
   'Maintenance Costs': [500, 700, 800, 1000, 1500, 1800, 2100, 2500],
   'Error Frequency': [1, 2, 1, 3, 2, 5, 3, 4}
df = pd.DataFrame(data)
corr = df.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
```

```
plt.title('Correlation Heatmap')
plt.show()
```

OUTPUT:



3.Multivariate Visualization:

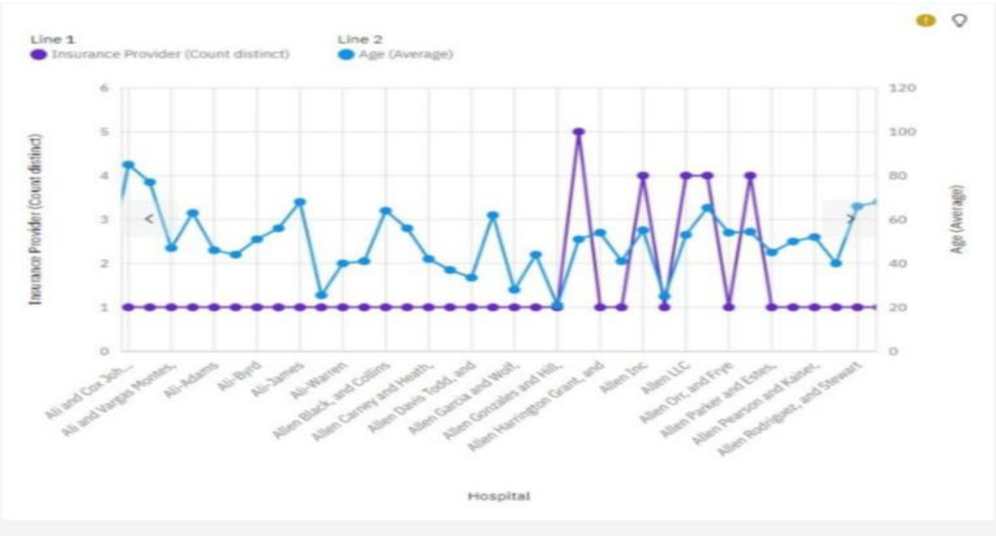
Multivariate visualization involves graphically representing relationships among three or more variables from the dataset. This technique provides a more comprehensive view of the interactions and dependencies among multiple factors, aiding in deeper insights and better decision-making.

CODE:

```
time_period = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug']
error_frequency = [1, 3, 2, 4, 3, 5, 2, 4]
plt.plot(time_period, error_frequency, marker='o', color='red')
plt.title('Error Frequency Over Time')
plt.xlabel('Time Period')
plt.ylabel('Error Frequency')
plt.grid(True)
```

```
plt.show()
```

OUTPUT:

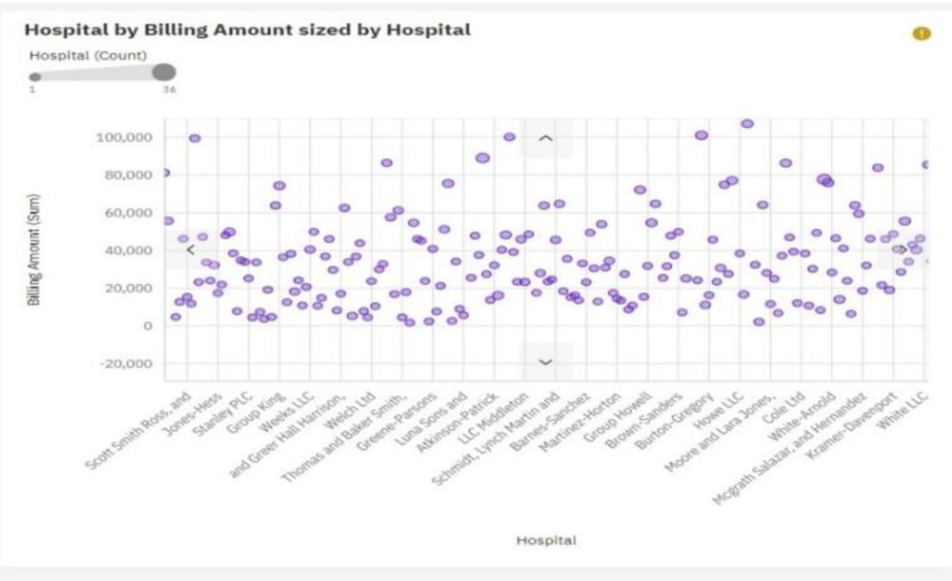


CODE:

```
import matplotlib.pyplot as plt
operating_hours = [100, 200, 300, 400, 500, 600, 700, 800]
maintenance_costs = [500, 700, 800, 1000, 1500, 1800, 2100, 2500]
error_frequency = [1, 2, 1, 3, 2, 5, 3, 4]
plt.scatter(operating_hours, maintenance_costs, s=[freq*100 for freq in error_frequency], alpha=0.5, c='blue')
plt.title('Bubble Chart of Operating Hours, Maintenance Costs, and Error Frequency')
plt.xlabel('Operating Hours')
plt.ylabel('Maintenance Costs')
for i in range(len(operating_hours)):
 plt.text(operating_hours[i], maintenance_costs[i], error_frequency[i], fontsize=9)
```

```
plt.show()
```

OUTPUT:



4.Interactive Visualization:

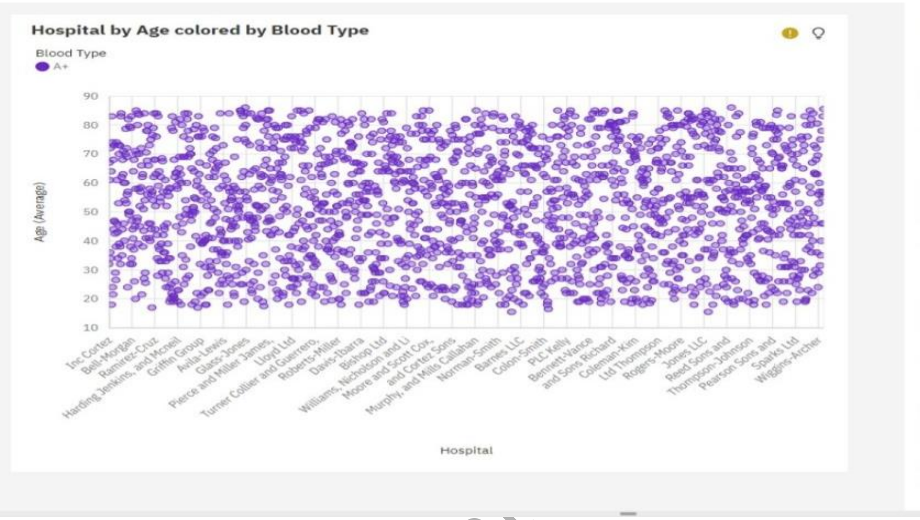
Interactive Scatterplot:

CODE:

```
import plotly.express as px

import pandas as pd

data = {

    'Operating Hours': [100, 200, 300, 400, 500, 600, 700, 800],

    'Maintenance Costs': [500, 700, 800, 1000, 1500, 1800, 2100, 2500],

    'Equipment Type': ['MRI', 'X-Ray', 'Ventilator', 'Infusion Pump', 'Ultrasound',
'MRI', 'Ventilator', 'X-Ray']}

df = pd.DataFrame(data)

fig = px.scatter(df, x='Operating Hours', y='Maintenance Costs',
```

```
              color='Equipment Type',
              hover_data=['Equipment Type'],
              title='Operating Hours vs. Maintenance Costs')
fig.show()
```

OUTPUT:



Interactive Dashboards:

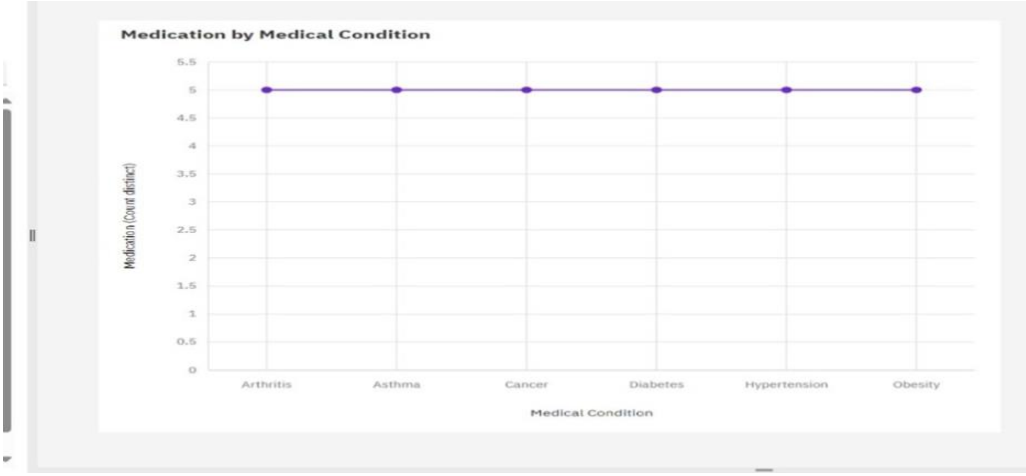
CODE:

```
import dash
from dash import dcc, html
from dash.dependencies import Input, Output
import plotly.express as px
import pandas as pd
app = dash.Dash(_name_)
data = {
```

```
    'Equipment ID': ['E1', 'E2', 'E3', 'E4', 'E5', 'E6', 'E7', 'E8'],
    'Equipment Type': ['MRI', 'X-Ray', 'Ventilator', 'Infusion Pump', 'Ultrasound', 'MRI', 'Ventilator', 'X-Ray'],
    'Operating Hours': [100, 200, 300, 400, 500, 600, 700, 800],
    'Maintenance Costs': [500, 700, 800, 1000, 1500, 1800, 2100, 2500],
    'Error Frequency': [1, 2, 1, 3, 2, 5, 3, 4],
    'Age of Equipment': [1, 2, 2, 3, 4, 5, 6, 7]}
df = pd.DataFrame(data)
app.layout = html.Div([
    html.H1("Predictive Maintenance Dashboard"),
    dcc.Dropdown(
        id='equipment-type-dropdown',
        options=[{'label': etype, 'value': etype} for etype in df['Equipment Type'].unique()],
        value='MRI',
        multi=True ),
    dcc.Graph(id='scatter-plot'),
    dcc.Graph(id='line-plot'),
    dcc.Graph(id='heatmap')])
@app.callback(
    Output('scatter-plot', 'figure'),
    Input('equipment-type-dropdown', 'value')
)
def update_scatter(selected_types):
    filtered_df = df[df['Equipment Type'].isin(selected_types)]
    fig = px.scatter(filtered_df, x='Operating Hours', y='Maintenance Costs',
```

```
                 color='Equipment Type', title='Operating Hours vs. Maintenance Costs')
    return fig
@app.callback(
    Output('line-plot', 'figure'),
    Input('equipment-type-dropdown', 'value')
)
def update_line(selected_types):
    filtered_df = df[df['Equipment Type'].isin(selected_types)]
    fig = px.line(filtered_df, x='Equipment ID', y='Error Frequency',
               color='Equipment Type', title='Error Frequency Over Time', markers=True)
    return fig
@app.callback(
    Output('heatmap', 'figure'),
    Input('equipment-type-dropdown', 'value')
)
def update_heatmap(selected_types):
    filtered_df = df[df['Equipment Type'].isin(selected_types)]
    corr = filtered_df.corr()
    fig = px.imshow(corr, text_auto=True, title='Correlation Heatmap')
    return fig
if _name_ == '_main_':
    app.run_server(debug=True)
```

OUTPUT:



Model development:

1.Data Collection:

Gather relevant data from the healthcare equipment, such as sensor readings, maintenance logs, environmental conditions, and historical failure data. Ensure the data is accurate, comprehensive, and properly labeled.

2.Data Preprocessing:

Clean the data by handling missing values, removing outliers, and normalizing or scaling numerical features. Also, feature engineering may be necessary to create new informative features from the raw data.

3.Feature Selection:

Identify the most relevant features that contribute to equipment failure prediction. Techniques such as correlation analysis, feature importance from machine learning models, or domain knowledge can guide feature selection.

4.Model Selection:

Choose appropriate machine learning algorithms or statistical methods based on the nature of the problem and the characteristics of the data. Consider using algorithms like regression, time series analysis, machine learning models (e.g., decision trees, random forests, gradient boosting), or deep learning models (e.g., neural networks).

5.Model Training:

Split the data into training and validation sets. Train the selected models on the training data and fine-tune hyperparameters to optimize performance. Cross-validation techniques can help assess model generalization.

6.Model Evaluation:

Evaluate the trained models using appropriate metrics such as accuracy, precision, recall, F1-score, or area under the ROC curve (AUC). Validate the models on the validation set to assess their performance and generalization ability.

7.Model Deployment:

Once a satisfactory model is developed, deploy it into the healthcare environment for real-time prediction. Integration with existing systems may be necessary to automate the maintenance scheduling based on model predictions.

Evaluation metrices:

1.Mean Time Between Failures (MTBF):

Measures the average time between equipment failures

2.Mean Time to Repair (MTTR):

Measures the average time taken to repair the equipment after a failure occurs.

3.Overall Equipment Effectiveness (OEE):

Evaluates the equipment's productivity, quality, and availability.

4.False Positive Rate:

Measures how often the system incorrectly predicts a failure when none occurs, impacting resource allocation and workflow disruptions.

5.True Positive Rate:

Measures how often the system correctly predicts a failure when one occurs, ensuring timely maintenance to prevent downtime.

6.Maintenance Cost:

Tracks the cost associated with performing predictive maintenance versus reactive maintenance or unplanned downtime.

7.Equipment Utilization:

Measures how effectively the equipment is being used over time.

8.Sensitivity and Specificity:

Assess the model's ability to correctly identify both positive (failure) and negative (normal operation) instances.

9.Precision and Recall:

Precision measures the accuracy of positive predictions, while recall measures the proportion of actual positives that were correctly identified.

Assumed Scenario:

In this scenario, we are managing the maintenance of healthcare equipment in a large hospital. The hospital has a wide range of medical devices including MRI machines, X-ray machines, ventilators, infusion pumps, and ultrasound machines. The goal is to predict maintenance needs and prevent equipment failures, thus ensuring the highest level of care for patients and minimizing downtime for critical equipment.

1. Equipment Inventory:

   - MRI Machines: 10 units
   - X-ray Machines: 15 units
   - Ventilators: 20 units
   - Infusion Pumps: 30 units
   - Ultrasound Machines: 12 units

2. Data Collection:

   - Operating Hours: The number of hours each equipment has been in operation.
   - Maintenance Costs: The costs incurred for maintaining the equipment.

- Error Frequency: The number of errors or faults reported by the equipment.
- Age of Equipment: The age of the equipment in years.
- Downtime: The number of hours the equipment was unavailable due to maintenance.

Conclusion:

The conclusion of a predictive maintenance project for healthcare equipment would typically summarize the findings and outcomes of the analysis. It would highlight the effectiveness of predictive models in identifying potential equipment failures before they occur, thus minimizing downtime and optimizing maintenance schedules. Additionally, it might emphasize the cost savings and improved patient care resulting from proactive maintenance strategies. Finally, it could discuss potential areas for future research or optimization to further enhance the reliability and efficiency of healthcare equipment maintenance.