**✅ Step 0: Setup**

1. Install required libraries:

bash

CopyEdit

pip install pandas numpy matplotlib seaborn scikit-learn streamlit openpyxl

1. Load your dataset:

python

CopyEdit

import pandas as pd

df = pd.read\_excel("robot\_logs.xlsx")

**🧹 Step 1: Data Preprocessing**

* ✅ Convert timestamp:

python

CopyEdit

df['timestamp'] = pd.to\_datetime(df['timestamp'])

* ✅ Handle missing values:

python

CopyEdit

df = df.dropna() # or use fillna()

* ✅ Create derived columns:

python

CopyEdit

df['battery\_drop'] = df['battery\_percent'].diff().fillna(0)

df['temp\_avg'] = df['cpu\_temp\_c'].rolling(3).mean().fillna(method='bfill')

* ✅ Encode categorical data:

python

CopyEdit

df['error\_code\_encoded'] = df['error\_code'].astype('category').cat.codes

**📊 Step 2: Exploratory Data Analysis (EDA)**

Use matplotlib or seaborn:

python

CopyEdit

import matplotlib.pyplot as plt

import seaborn as sns

# Battery % over time

df.plot(x='timestamp', y='battery\_percent', title='Battery Over Time')

# CPU temp histogram

sns.histplot(df['cpu\_temp\_c'])

# Error count

sns.countplot(data=df, x='error\_code')

# Correlation matrix

sns.heatmap(df.corr(numeric\_only=True), annot=True)

**🛠️ Step 3: Feature Engineering**

Add additional features:

python

CopyEdit

df['hour'] = df['timestamp'].dt.hour

df['is\_low\_battery'] = df['battery\_percent'] < 20

df['task\_load\_avg'] = df['task\_load'].rolling(5).mean().fillna(method='bfill')

**🤖 Step 4: Model Building**

Split data:

python

CopyEdit

from sklearn.model\_selection import train\_test\_split

features = ['battery\_percent', 'cpu\_temp\_c', 'motor\_current\_a', 'task\_load']

X = df[features]

y = df['error\_code\_encoded']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

Train a classifier:

python

CopyEdit

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()

model.fit(X\_train, y\_train)

**📈 Step 5: Model Evaluation**

python

CopyEdit

from sklearn.metrics import classification\_report, confusion\_matrix

y\_pred = model.predict(X\_test)

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

**📉 Step 6: Regression (Optional)**

For predicting battery life or temperature rise:

python

CopyEdit

from sklearn.linear\_model import LinearRegression

reg = LinearRegression()

reg.fit(X\_train, df['battery\_percent'])

Evaluate:

python

CopyEdit

from sklearn.metrics import mean\_squared\_error

y\_reg\_pred = reg.predict(X\_test)

print(mean\_squared\_error(df['battery\_percent'], y\_reg\_pred, squared=False)) # RMSE

**🌐 Step 7: Streamlit Dashboard (Optional)**

Create dashboard.py:

python

CopyEdit

import streamlit as st

st.title("Robot Health Dashboard")

st.line\_chart(df.set\_index('timestamp')['battery\_percent'])

st.bar\_chart(df['error\_code'].value\_counts())

st.write("Latest CPU Temp:", df['cpu\_temp\_c'].iloc[-1])

Run it:

bash

CopyEdit

streamlit run dashboard.py

**🧾 Step 8: Documentation**

Include in your report:

* Project goal
* Dataset structure and simulation logic
* EDA findings (plots)
* Modeling approach and accuracy
* Dashboard screenshots (if built)
* Suggestions for future improvements

**🔮 Step 9: Future Work (Optional)**

* Time series forecasting with Prophet or LSTM
* Real-time sensor data ingestion
* Use Reinforcement Learning for self-healing logic
* NLP on error logs

**✅ Summary Checklist**

| **Step** | **Task** | **Status** |
| --- | --- | --- |
| 0 | Setup & Load Data | ✅ |
| 1 | Preprocessing | ⬜ |
| 2 | EDA | ⬜ |
| 3 | Feature Engineering | ⬜ |
| 4 | Train Classifier | ⬜ |
| 5 | Evaluate Model | ⬜ |
| 6 | Regression (Optional) | ⬜ |
| 7 | Dashboard (Optional) | ⬜ |
| 8 | Report | ⬜ |
| 9 | Future Enhancements | ⬜ |