pip install flask scikit-learn pandas numpy rasa openai

from flask import Flask, request, jsonify

import numpy as np

import pandas as pd

import joblib

from datetime import datetime

import openai

# Load predictive maintenance model

model = joblib.load("predictive\_maintenance\_model.pkl")

# OpenAI API key

openai.api\_key = "your\_openai\_api\_key"

# Flask app

app = Flask(\_name\_)

# Endpoint for predictive maintenance

@app.route("/predict", methods=["POST"])

def predict():

try:

# Get sensor data from request

data = request.json

sensors = np.array(data["sensors"]).reshape(1, -1)

# Make a prediction

prediction = model.predict(sensors)[0]

probability = model.predict\_proba(sensors)[0]

# Format the response

response = {

"prediction": "Failure" if prediction == 1 else "No Failure",

"failure\_probability": f"{probability[1]:.2f}",

"timestamp": datetime.now().isoformat()

}

return jsonify(response)

except Exception as e:

return jsonify({"error": str(e)})

# Endpoint for chatbot integration

@app.route("/chatbot", methods=["POST"])

def chatbot():

try:

# Get user query

user\_query = request.json.get("query", "")

# Use OpenAI GPT to process the query

response = openai.ChatCompletion.create(

model="gpt-3.5-turbo",

messages=[

{"role": "system", "content": "You are a maintenance chatbot that provides predictive maintenance insights."},

{"role": "user", "content": user\_query},

],

)

chatbot\_response = response["choices"][0]["message"]["content"]

return jsonify({"response": chatbot\_response})

except Exception as e:

return jsonify({"error": str(e)})

if \_name\_ == "\_main\_":

app.run(debug=True)

import pandas as pd

import numpy as np

from sklearn.ensemble import RandomForestClassifier

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

from sklearn.metrics import accuracy\_score

import joblib

# Sample dataset

# Replace with actual sensor data

data = pd.DataFrame({

"sensor1": np.random.rand(100),

"sensor2": np.random.rand(100),

"sensor3": np.random.rand(100),

"failure": np.random.choice([0, 1], size=100)

})

# Features and target

X = data[["sensor1", "sensor2", "sensor3"]]

y = data["failure"]

# Split data

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

# Train model

model = RandomForestClassifier()

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

# Evaluate model

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

print(f"Accuracy: {accuracy\_score(y\_test, y\_pred):.2f}")

# Save the model

joblib.dump(model, "predictive\_maintenance\_model.pkl")

python app.py

curl -X POST -H "Content-Type: application/json" \

-d '{"sensors": [0.5, 0.7, 0.3]}' \

http://127.0.0.1:5000/predict

curl -X POST -H "Content-Type: application/json" \

-d '{"query": "What is the machine failure probability?"}' \

http://127.0.0.1:5000/chatbot