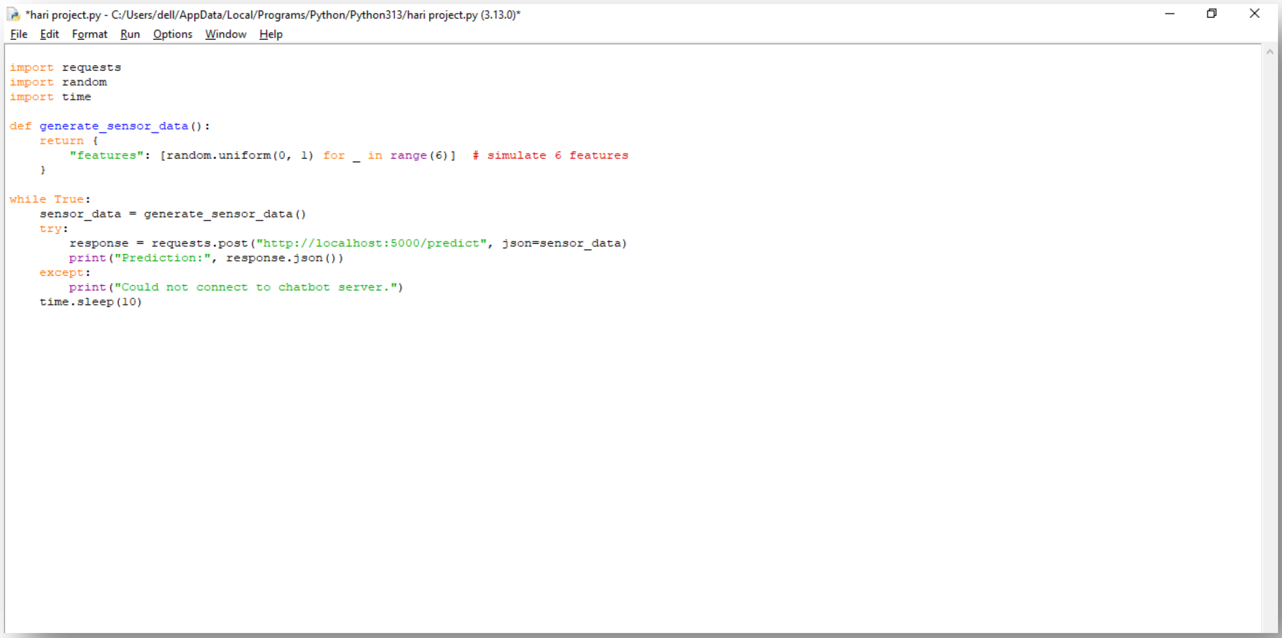
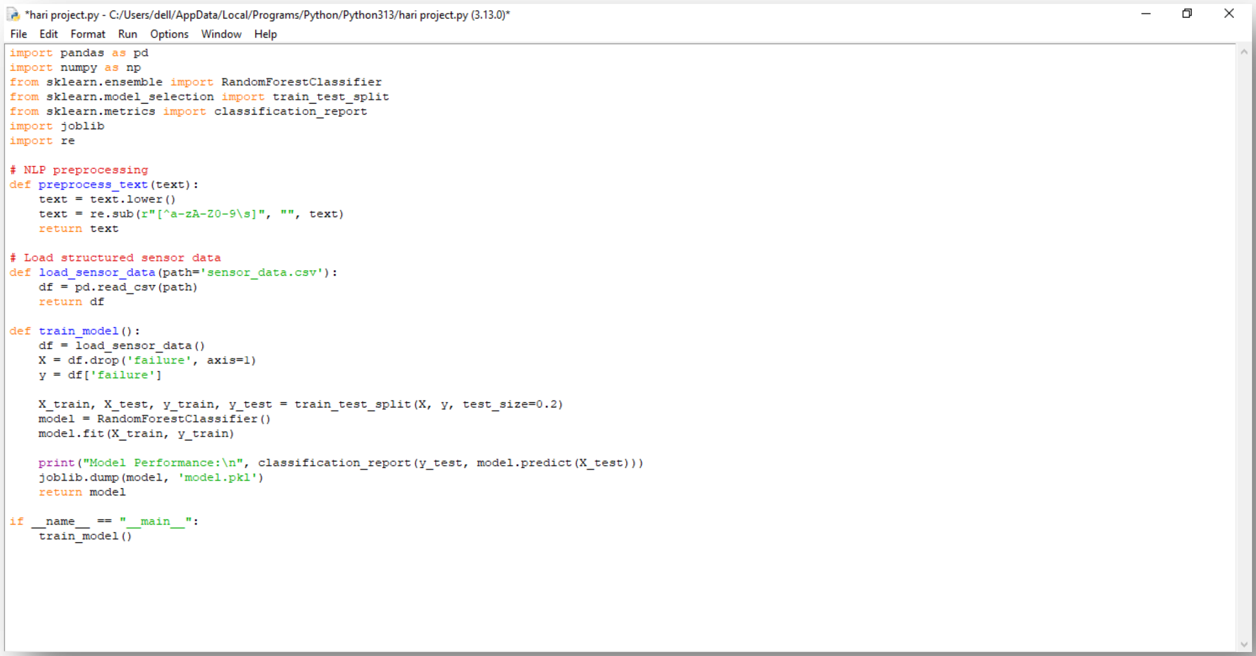
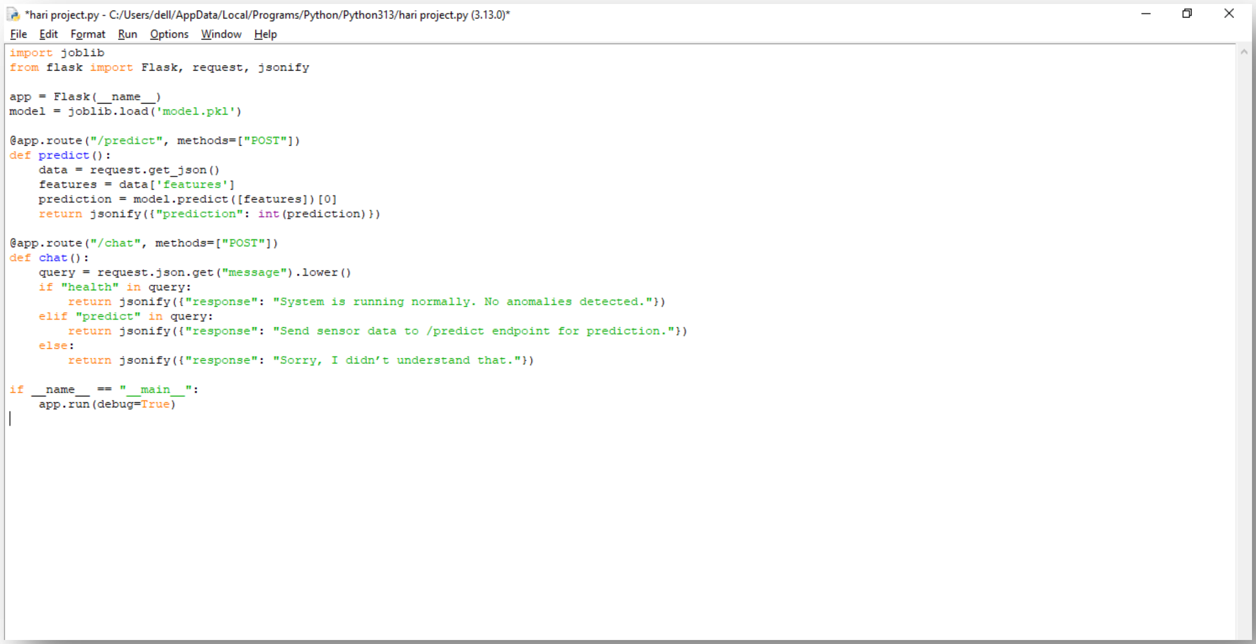

hari project.py - C:/Users/dell/AppData/Local/Programs/Python/Python313/hari project.py (3.13.0)
File Edit Format Run Options Window Help

```python
import requests
import random
import time

def generate_sensor_data():
    return {
        "features": [random.uniform(0, 1) for _ in range(6)]  # simulate 6 features
    }

while True:
    sensor_data = generate_sensor_data()
    try:
        response = requests.post("http://localhost:5000/predict", json=sensor_data)
        print("Prediction:", response.json())
    except:
        print("Could not connect to chatbot server.")
    time.sleep(10)
```


hari project.py - C:/Users/dell/AppData/Local/Programs/Python/Python313/hari project.py (3.13.0)
File Edit Format Run Options Window Help

```python
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 classification_report
import joblib
import re

# NLP preprocessing
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r"[^a-zA-Z0-9\s]", "", text)
    return text

# Load structured sensor data
def load_sensor_data(path='sensor_data.csv'):
    df = pd.read_csv(path)
    return df

def train_model():
    df = load_sensor_data()
    X = df.drop('failure', axis=1)
    y = df['failure']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
    model = RandomForestClassifier()
    model.fit(X_train, y_train)

    print("Model Performance:\n", classification_report(y_test, model.predict(X_test)))
    joblib.dump(model, 'model.pkl')
    return model

if __name__ == "__main__":
    train_model()
```


hari project.py - C:/Users/dell/AppData/Local/Programs/Python/Python313/hari project.py (3.13.0)
File Edit Format Run Options Window Help

```python
import joblib
from flask import Flask, request, jsonify

app = Flask(__name__)
model = joblib.load('model.pkl')

@app.route("/predict", methods=["POST"])
def predict():
    data = request.get_json()
    features = data['features']
    prediction = model.predict([features])[0]
    return jsonify({"prediction": int(prediction)})

@app.route("/chat", methods=["POST"])
def chat():
    query = request.json.get("message").lower()
    if "health" in query:
        return jsonify({"response": "System is running normally. No anomalies detected."})
    elif "predict" in query:
        return jsonify({"response": "Send sensor data to /predict endpoint for prediction."})
    else:
        return jsonify({"response": "Sorry, I didn’t understand that."})

if __name__ == "__main__":
    app.run(debug=True)
```