Code for major project. This code includes machine learning as well as connection with thingspeak and getting data continuously.

import pandas as pd

import numpy as np

import pickle

import requests

import streamlit as st

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

from sklearn.ensemble import RandomForestClassifier

# 1. THINGSPEAK DETAILS

THING\_SPEAK\_CHANNEL\_ID = "2849586"   # Example Channel ID

READ\_API\_KEY = "XVZ5FB3IFVVZ6Y2"     # Example Read API Key

# Fetch only the latest entry from the channel:

THING\_SPEAK\_URL = f"https://api.thingspeak.com/channels/{THING\_SPEAK\_CHANNEL\_ID}/feeds.json?api\_key={READ\_API\_KEY}&results=1"

# 2. DATASET LOADING & PREPROCESSING

@st.cache\_data

def load\_data():

    """

    Reads the CSV file containing cattle health data.

    Returns a DataFrame.

    """

    df = pd.read\_csv("synthetic\_cattle\_health\_data\_3.csv")

    return df

def preprocess\_data(df):

    """

    Splits the data into features (X) and target (y),

    then performs a train-test split.

    """

    # Features: sensor readings

    X = df[['Heart\_Rate\_bpm', 'Temperature\_C', 'Humidity\_percent', 'Respiratory\_Rate\_breaths\_min']]

    # Target: disease/health status

    y = df['Health\_Status']

    # Split data into train (80%) and test (20%)

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

    return X\_train, X\_test, y\_train, y\_test

# 3. MODEL TRAINING & SAVING

def train\_and\_save\_model(X\_train, y\_train):

    """

    Trains a RandomForestClassifier on the training data,

    then saves the trained model to 'model.pkl'.

    """

    model = RandomForestClassifier(n\_estimators=100, random\_state=42)

    model.fit(X\_train, y\_train)

    with open("model.pkl", "wb") as f:

        pickle.dump(model, f)

# 4. MODEL LOADING

def load\_model():

    """

    Loads the trained model from 'model.pkl' for prediction.

    """

    with open("model.pkl", "rb") as f:

        return pickle.load(f)

# 5. FETCH LATEST SENSOR DATA FROM THINGSPEAK

def fetch\_sensor\_data():

    """

    Sends a GET request to ThingSpeak to fetch the latest sensor data.

    Returns a NumPy array shaped (1,4) if successful, else None.

    """

    response = requests.get(THING\_SPEAK\_URL)

    if response.status\_code == 200:

        data = response.json()

        feeds = data.get("feeds", [])

        if feeds:

            latest\_entry = feeds[-1]

            try:

                # Convert each field to float, then to int if needed

                heart\_rate    = float(latest\_entry["field1"])

                temperature   = float(latest\_entry["field2"])

                humidity      = float(latest\_entry["field3"])

                respiratory   = float(latest\_entry["field4"])

                # Reshape into 2D array for model prediction

                return np.array([[heart\_rate, temperature, humidity, respiratory]])

            except (ValueError, TypeError):

                # If fields are empty or invalid

                return None

    return None

# 6. PREDICTION LOGIC

def predict\_disease(model, input\_data):

    """

    Uses the loaded model to predict disease given sensor input\_data.

    Returns the predicted label as a string.

    """

    prediction = model.predict(input\_data)

    return prediction[0]  # Extract the label

# 7. STREAMLIT APP

def main():

    st.title("🐄 Automated Cow Disease Prediction System")

    st.markdown("This app \*\*automatically\*\* fetches the latest sensor data from ThingSpeak and predicts the cow's health status using a RandomForest model.")

    # 7.1. Fetch latest data from ThingSpeak

    sensor\_data = fetch\_sensor\_data()

    if sensor\_data is not None:

        # 7.2. Load the trained model

        model = load\_model()

        # 7.3. Predict disease

        disease\_prediction = predict\_disease(model, sensor\_data)

        # 7.4. Display fetched data and prediction

        st.subheader("Latest Sensor Data from ThingSpeak")

        st.write(f"- \*\*Heart Rate (bpm)\*\*: {sensor\_data[0][0]}")

        st.write(f"- \*\*Temperature (°C)\*\*: {sensor\_data[0][1]}")

        st.write(f"- \*\*Humidity (%)\*\*: {sensor\_data[0][2]}")

        st.write(f"- \*\*Respiratory Rate\*\*: {sensor\_data[0][3]}")

        st.subheader("Predicted Health Status")

        st.write(f"\*\*{disease\_prediction}\*\*")

    else:

        st.error("Failed to fetch valid data from ThingSpeak. Please check your Read API Key, Channel ID, and fields.")

# 8. MAIN EXECUTION

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

    # Load and preprocess data

    df = load\_data()

    X\_train, X\_test, y\_train, y\_test = preprocess\_data(df)

    # Train model if 'model.pkl' doesn't exist

    try:

        open("model.pkl", "rb").close()

    except FileNotFoundError:

        train\_and\_save\_model(X\_train, y\_train)

    # Run the Streamlit app

    main()