### **Objective**

Develop a machine learning pipeline to predict mycotoxin levels (DON concentration) in corn samples using hyperspectral imaging data. This assignment tests your ability to preprocess complex data, build and optimize regression models, and design code that is modular and production-ready.

## **Problem Statement**

You are provided with a compact hyperspectral dataset containing spectral reflectance data of corn samples across multiple wavelength bands. Each row represents a corn sample, and the features are the reflectance values at different wavelengths. The target variable is the DON concentration (a continuous variable).

Your goal is to:

1. **Preprocess** the data (handle missing values, normalize features, and explore potential anomalies).
2. **Visualize** the spectral bands to understand data characteristics.
3. **Train** a regression model (e.g., a neural network) to predict DON concentration.
4. **Evaluate** the model using robust metrics and visualization tools.
5. **Develop a production-ready pipeline** with modular code, testing, and documentation.

## **Dataset Description**

* **Features:** Spectral reflectance values measured at various wavelengths.
* **Samples:** Each row represents an individual corn sample.
* **Target:** DON concentration (continuous numerical value).

## **Tasks & Guidelines**

### **1. Data Exploration and Preprocessing**

* **Loading & Inspection:**
  + Load the dataset and inspect for missing values, outliers, and inconsistencies.
  + Provide summary statistics and visualizations (e.g., histograms, boxplots).
* **Preprocessing:**
  + Handle missing data (imputation strategies or removal).
  + Normalize or standardize the spectral data.
  + Detect and, if necessary, remove or flag anomalies.
  + Generate visualizations:
    - Line plots for average reflectance over wavelengths.
    - Heatmaps or pairplots for sample comparisons.
* **Advanced Data Quality Checks:**
  + Automate checks for sensor drift or data inconsistencies.
  + Optionally create additional features (e.g., spectral indices) that may enhance prediction.

### **2. Model Training**

* **Model Selection:**
  + Choose a baseline regression model. A simple neural network is recommended.
  + Optionally explore advanced models such as ensemble methods (stacking/boosting) or custom architectures.
* **Data Splitting:**
  + Split the data into training (80%) and testing (20%) sets.
  + Consider implementing k-fold cross-validation for a robust evaluation.
* **Hyperparameter Optimization:**
  + Optimize hyperparameters using grid search, random search, or Bayesian Optimization (e.g., with Optuna).
  + If applicable, experiment with a custom loss function that reflects the domain needs.
* **Production Readiness:**
  + Structure your code in a modular fashion (e.g., separate modules for data processing, modeling, and evaluation).
  + Include clear documentation and type annotations.

### **3. Model Evaluation**

* **Metrics:**
  + Calculate regression metrics:
    - Mean Absolute Error (MAE)
    - Root Mean Squared Error (RMSE)
    - R² Score
* **Visual Evaluation:**
  + Plot a scatter plot comparing actual vs. predicted values.
  + Perform residual analysis to identify any systematic errors.
* **Interpretability:**
  + Use interpretability tools like SHAP or LIME to explain predictions.
  + Summarize feature importance and discuss model limitations.

### **4. Pipeline Integration and Production-Readiness**

* **Code Quality:**
  + Write unit tests for core functionalities.
  + Use logging to capture runtime details and potential issues.
* **Deployment:**
  + Package your model as a Docker container or deploy it via an API using Flask/FastAPI.
  + Ensure your pipeline can accept new spectral data for on-the-fly predictions.

## **Deliverables**

1. **GitHub Repository** The repository should include:  
   * **Jupyter Notebook or Python Script:**
     + Modular, clean, and well-commented code that covers all tasks.
   * **Short Report (1–2 pages):**
     + Outline preprocessing steps and rationale.
     + Summarize insights from dimensionality reduction.
     + Detail model selection, training, and evaluation.
     + Highlight key findings and suggest possible improvements.
   * **README File:**
     + Provide setup instructions, dependency installation, and how to run the code.
     + Describe the repository structure.
   * **Deployment Scripts/Instructions:**
     + Dockerfile and/or API service code (e.g., Flask/FastAPI).
   * **Documentation:**
     + Detailed inline documentation and external docs if applicable.
2. **Submission:**
   * Fill this form - https://forms.gle/bX52TYJRe2cYvDbF6

## **Evaluation Criteria**

* **Code Quality (50%):**
  + Clean, organized, and well-documented code.
  + Modular design that facilitates integration and future expansion.
* **EDA & Visualization (10%):**
  + Thorough exploration and clear visualizations.
  + Effective handling of data anomalies and preprocessing.
* **Model Performance (10%):**
  + Appropriate model choice and robust evaluation.
  + Evidence of hyperparameter tuning and performance analysis.
* **Interpretability & Production-Readiness (30%):**
  + Use of interpretability methods (SHAP, LIME, etc.).
  + Implementation of unit tests, logging, and deployment readiness.

## **Bonus (Optional)**

* **Attention Mechanism/Transformer:**
  + Implement and compare an attention-based model to the baseline.
* **Interactive App:**
  + Create a Streamlit app to allow users to upload spectral data and receive predictions in real time.
* **Ensemble Methods:**
  + Experiment with ensemble techniques (e.g., stacking multiple models) and report performance improvements.
* **CI/CD Integration:**
  + Set up continuous integration tests with automated deployment pipelines.

## **Submission Guidelines**

* **Deadline:** 11:59 PM, March 25, 2025
* **Submission:** Fill this form - https://forms.gle/bX52TYJRe2cYvDbF6