Corn Mycotoxin Prediction using Hyperspectral Imaging

1. Introduction

This project focuses on predicting mycotoxin (**DON**, short for *deoxynivalenol*) levels in corn samples using hyperspectral imaging data. DON is a toxic compound produced by fungi that can contaminate crops and pose serious health risks. With hundreds of spectral features available per sample, the objective was to clean the dataset, explore and visualize patterns, train a deep learning model to predict DON levels, and create a user-friendly web app using Streamlit.

2. Data Preprocessing

The raw dataset included:

- `hsi_id`: A unique identifier for each corn sample
- Hundreds of spectral reflectance features (wavelength bands)
- `vomitoxin_ppb`: The DON concentration (target variable)

Code Highlights:

- We dropped `hsi_id` because it's not useful for modeling.
- Used StandardScaler to normalize all spectral bands, ensuring they have a mean of 0 and standard deviation of 1. This step is essential for neural networks and PCA/t-SNE to function effectively.

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Justification:

> Scaling was crucial because reflectance values across wavelengths differ in range, and deep learning models perform better with normalized input. This also improved the clarity of dimensionality reduction plots.

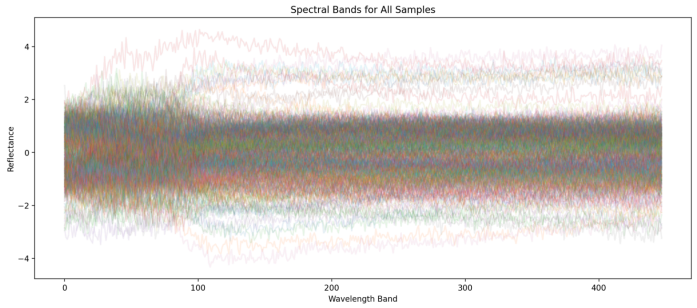
Challenge:

Handling high dimensionality and ensuring consistent preprocessing between training and prediction phases.

3. Data Visualization & Exploration

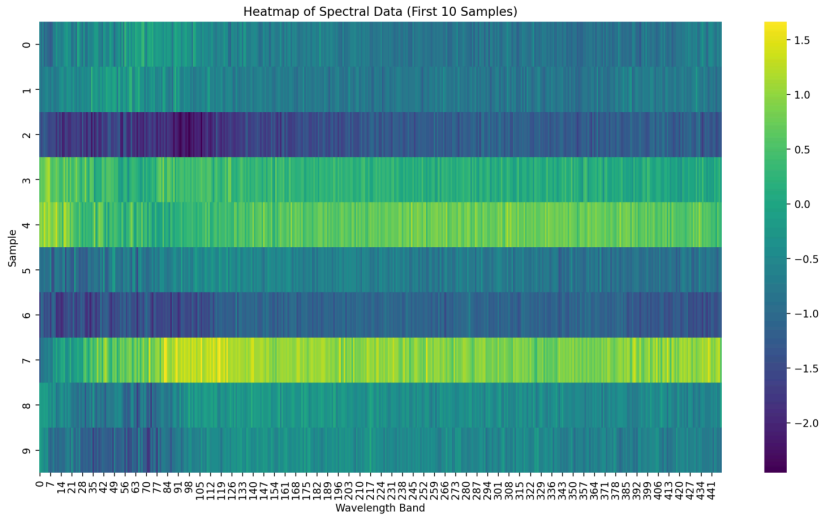
Spectral Band Overview

- We generated line plots for every sample to understand spectral variations across bands



- It shows most samples cluster around a baseline, but some show prominent peaks or dips. This variation is potentially linked to DON concentration.

Heatmap of First 10 Samples



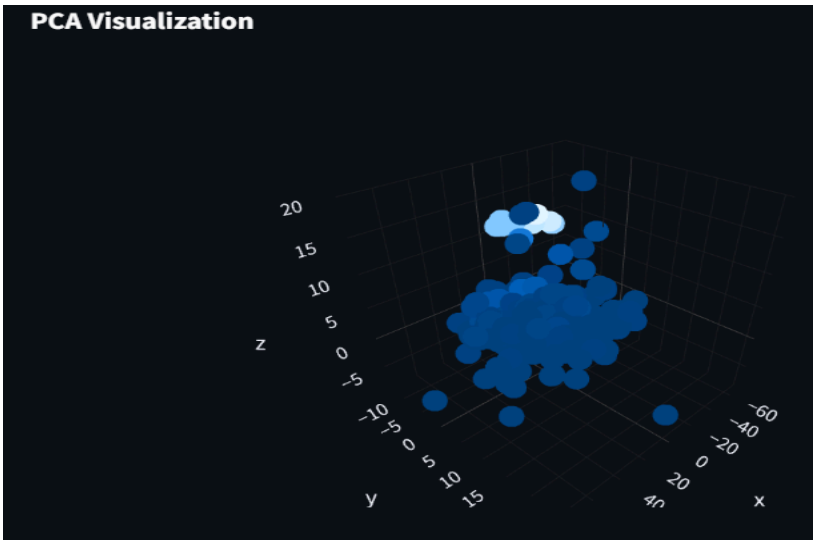
This heatmap highlights band intensity across 10 samples. Samples 3 and 7 show distinctly brighter/darker patterns, which could suggest higher/lower toxin levels.

These plots helped validate that the spectral data was meaningful and varied enough for modeling.

4. Dimensionality Reduction

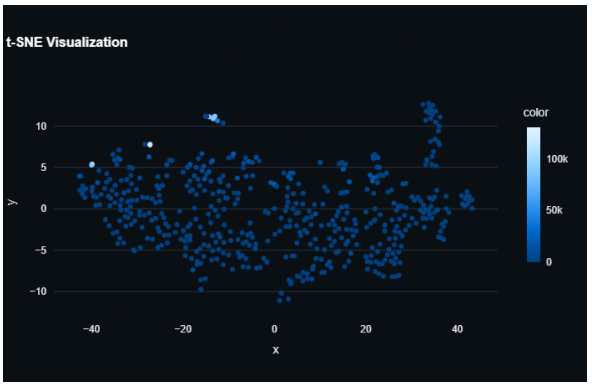
Dimensionality reduction techniques were used to visualize high-dimensional spectral data in fewer dimensions.

PCA (Principal Component Analysis)



PCA simplified the data into 3 principal components. We observed that samples formed visible clusters, indicating the reflectance features carry structured information useful for modeling.

T-SNE (t-distributed Stochastic Neighbor Embedding)



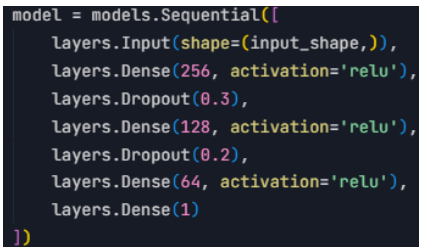
T-SNE helped visualize the local and global structure of the dataset. The color-coded clusters show how some samples group closely together—again hinting at patterns linked to DON.

Trade-off: PCA is interpretable but linear; t-SNE captures non-linear patterns but lacks interpretability.

5. Model Building & Training

We used a simple yet effective **feedforward neural network** (dense layers) to predict DON levels.

Code Summary:



```
model = models.Sequential([
    layers.Input(shape=(input_shape,)),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(64, activation='relu'),
    layers.Dense(1)
])
```

Explanation:

- We used multiple dense layers with ReLU activation.
- Dropout layers help prevent overfitting.
- The final output layer predicts a single DON value.
- The model was compiled with the Adam optimizer and MSE loss.

The model was trained on 80% of the data and validated on 20%. The evaluation was based on

- MAE (Mean Absolute Error)
- RMSE (Root Mean Squared Error)
- R^2 Score (how well predictions fit the real data)

Challenge: Avoiding overfitting with limited data while still capturing complex relationships.

Trade-off: A simple model is easier to train and tune but may lack precision compared to advanced architectures like transformers.

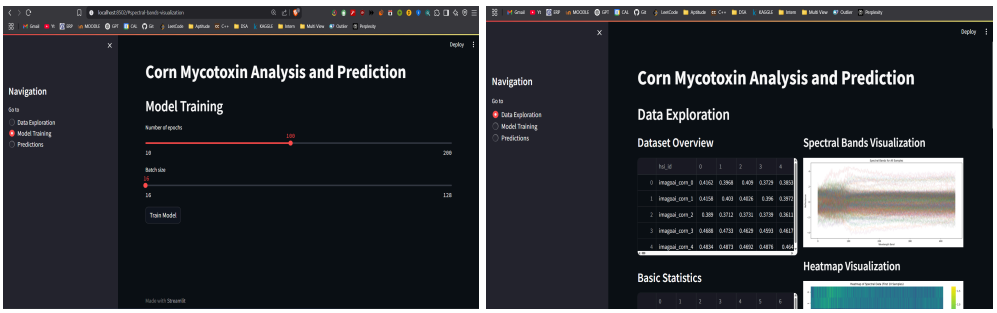
6. Interactive Streamlit Application

To make this project accessible, I built a web app using Streamlit.

Key Features:

- Visual exploration: spectral line plots, heatmaps, PCA, t-SNE
- Interactive model training: sliders for epochs and batch size
- Upload new data (CSV) and get instant DON predictions
- View model performance visually: loss curves and prediction accuracy

Streamlit made it easy to display plots, interact with model parameters, and upload files for predictions all in the browser.



The app allows even non-coders to explore and work with hyperspectral data.

7. Virtual Environment Setup

To keep the project reproducible and avoid version conflicts, I used a **virtual environment**.

Setup Steps:

```
# Create environment
python -m venv venv

# Activate it
venv\Scripts\activate  # On Windows

# Install dependencies
pip install -r requirements.txt
```

requirements.txt:

```
pandas==2.1.1
numpy==1.24.3
matplotlib==3.8.0
seaborn==0.12.2
scikit-learn==1.3.0
tensorflow==2.14.0
plotly==5.17.0
streamlit==1.27.2
scipy==1.11.2
joblib==1.3.2
h5py==3.9.0
tenacity==8.2.3
```

This setup avoids version mismatches and missing dependencies.