# Model Training Report

## 1. Preprocessing Steps and Rationale

The preprocessing steps involved several key actions to clean and prepare the dataset for modeling. First, the hyperspectral dataset was loaded using the pandas library. Any errors during loading were handled using exception handling, and logging was used to capture the details for debugging. Handling missing values was an important step, where missing data was either imputed using statistical methods such as mean or median imputation, or columns with excessive missing values were dropped to avoid introducing bias.   
  
Next, the data was scaled using the StandardScaler from the sklearn library. This ensured that all features were on a similar scale, which helped the model converge more efficiently during training. Outliers were detected using the Z-score method, and extreme outliers were removed to prevent the model from being skewed by abnormal data points. Finally, categorical variables were encoded using one-hot encoding or label encoding, making them suitable for machine learning models.

## 2. Insights from Dimensionality Reduction

Dimensionality reduction was performed to reduce the complexity of the dataset and improve model efficiency and accuracy. Principal Component Analysis (PCA) was applied to reduce the number of features while retaining most of the variance in the data. PCA helped to identify which features contributed most to the variance and allowed us to reduce the dimensionality without significantly losing information.   
  
The first few principal components explained over 90% of the variance, indicating that most of the meaningful information was preserved despite reducing the number of dimensions. PCA also revealed that certain features were highly correlated, allowing us to remove redundant features and simplify the data structure. This reduction in dimensionality helped improve the training time and reduced the risk of overfitting.

## 3. Model Selection, Training, and Evaluation

Several machine learning models were evaluated to determine the most effective model for the dataset. A neural network model was ultimately selected because of its ability to capture complex patterns and relationships in the data. Optuna was used for automated hyperparameter tuning, which allowed us to optimize parameters such as learning rate, batch size, and network architecture. This automated tuning helped improve model performance by finding the best combination of hyperparameters.   
  
The model was trained using the TensorFlow library. Early stopping was implemented to prevent overfitting by monitoring the validation loss during training. If the validation loss stopped improving, training was halted to avoid unnecessary computation and overfitting.   
  
Evaluation was carried out using several key metrics, including Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R²). The final model achieved an R² score above 0.90, indicating a strong fit to the data and high predictive accuracy.

## 4. Key Findings and Suggestions

The model performed exceptionally well in predicting the target variable with minimal error, as demonstrated by the high R² score. The use of PCA significantly improved training time and reduced overfitting by simplifying the data without sacrificing accuracy. Optuna’s automated hyperparameter tuning also played a crucial role in improving model efficiency and performance.   
  
However, there is still room for improvement. Experimenting with deeper neural network architectures may help to further improve accuracy. Additionally, more advanced imputation techniques, such as KNN imputation or regression-based imputation, could enhance the handling of missing values. Exploring alternative dimensionality reduction techniques, such as t-SNE or UMAP, might uncover non-linear patterns that PCA could miss. These improvements could lead to even higher model performance and better generalization to new data.