Model Construction and Comparisons

Random Forest (Supervised): The Random Forest classifier was constructed using scikit-learn. This ensemble method consists of multiple decision trees, which are trained on different subsets of the dataset and combined to produce a robust final prediction. Class weights were used to handle class imbalance. Pre-tuning involved setting the number of trees and depth, while hyperparameter tuning was performed using GridSearchCV to optimize parameters such as the number of estimators and maximum depth.

K-means Clustering (Unsupervised): For the K-means clustering model, features were standardized before clustering into 6 clusters, corresponding to the 6 spectral classes. The elbow method and silhouette scores were used to determine the optimal number of clusters. Post-clustering, the clusters were mapped to actual classes using majority voting to evaluate clustering accuracy. Hyperparameter tuning involved adjusting the number of clusters and initialization methods.

Multi-Layer Perceptron (MLP) (Supervised): The MLP model was built using PyTorch. It consisted of an input layer, one hidden layer with 128 neurons, and an output layer with 6 neurons. ReLU activation functions and dropout layers were used for regularization. Pre-tuning involved setting initial layer sizes and learning rates. SMOTE was applied to balance the class distribution. Hyperparameter tuning, conducted via GridSearchCV, optimized the number of neurons, dropout rates, and learning rates.

Convolutional Neural Network (CNN) (Supervised): A CNN model was constructed using PyTorch for structured data classification. Input data was reshaped and padded. The network included two convolutional layers, batch normalization, max pooling, dropout for regularization, and two fully connected layers. SMOTE was used for data balancing, and class weights were incorporated into the loss function. Pre-tuning involved initial layer configuration and learning rates. Hyperparameter tuning, done through GridSearchCV, optimized filter sizes, dropout rates, and learning rates.

Model Comparisons

The models were evaluated based on accuracy, precision, recall, and F1 score. The Random Forest model demonstrated balanced performance due to class weights. K-means clustering provided moderate accuracy, with clusters mapped to actual classes for evaluation. The MLP and CNN models showed improved performance for minority classes, benefiting from SMOTE and class weights. Hyperparameter tuning across models aimed to refine performance further, with the MLP and CNN models particularly benefiting from optimized parameters to enhance their ability to handle class imbalance and improve metrics.

By leveraging techniques like SMOTE, class weights, and comprehensive hyperparameter tuning, we enhanced the models' performance, particularly for minority classes, addressing class imbalance challenges effectively.