***M3 - Hyperparameter Tuning and Model Packaging Report***

**1)Hyperparameter Tuning Results**

**Objective:**

The goal was to optimize a RandomForestClassifier model on the Wine dataset from scikit-learn using Optuna, a framework for automated hyperparameter optimization.

**Dataset:**

The Wine dataset is a multiclass classification dataset with 13 features and 3 target classes. It contains data about chemical composition and sensory characteristics of wine.

**Optimization Approach:**

* **Library Used**: Optuna
* **Hyperparameters Tuned**:
  + n\_estimators: Number of trees in the forest (range: 10 to 200)
  + max\_depth: Maximum depth of each tree (range: 2 to 32, log-scaled)
  + min\_samples\_split: Minimum samples required to split an internal node (range: 2 to 20)

**Results:**

* **Total Trials Conducted**: 49
* **Best Trial**: Trial 36
* **Best Parameters**:
  + n\_estimators: 31
  + max\_depth: 17
  + min\_samples\_split: 10
* **Best Cross-Validation Accuracy**: 99.29%

**Final Model:**

The RandomForestClassifier was trained on the training dataset using the best hyperparameters and saved as best\_model.pkl using Joblib.

**Summary of Work Completed**

**1. Hyperparameter Tuning**

1. **Dataset Preparation**:
   * Loaded the Wine dataset from scikit-learn and split it into training and testing sets.
2. **Optimization with Optuna**:
   * Defined an objective function that performed 3-fold cross-validation on the training set for each trial.
   * Used Optuna to search for the best hyperparameters across 49 trials.
   * Saved the best-performing model for deployment.

**2. Model Packaging**

1. **Flask Application**:
   * Created a RESTful API using Flask to serve the model.
   * Implemented an endpoint (/predict) to accept input features as JSON and return predictions.
2. **Dockerization**:
   * Packaged the Flask application into a Docker container for ease of deployment and platform independence.
   * Created a Dockerfile to set up the Python environment and expose the API.

**3. Testing:**

* Ran the Docker container and tested the API using Thunderclient in VS code to ensure correct predictions.

**Justification for Choices Made**

**1. Optuna for Hyperparameter Tuning**

* **Reason**: Optuna is efficient for automated hyperparameter optimization and supports advanced search algorithms like Tree-structured Parzen Estimator (TPE). It allowed us to focus on maximizing accuracy with minimal manual intervention.
* **Result**: The best parameters achieved a significant cross-validation accuracy of 99.29%.

**2. Flask for Model Deployment**

* **Reason**: Flask is lightweight, easy to use, and ideal for creating REST APIs for machine learning models. It allows seamless integration of the model into a web service accessible via HTTP requests.
* **Result**: A simple API was created to make the model available for predictions.

**3. Docker for Containerization**

* **Reason**: Docker provides an isolated and reproducible environment, ensuring that the application runs consistently across different systems. It also simplifies deployment and scaling.
* **Result**: The model and Flask application were packaged into a Docker container for platform-independent deployment.

## ****Key Deliverables****

1. **Best Hyperparameters**:- Report in 1st page.
   * n\_estimators: 31, max\_depth: 17, min\_samples\_split: 10
   * Best cross-validation accuracy: 99.29%
2. **Artifacts**:- Attached in Folder.
   * Trained model: best\_model.pkl
   * Flask application: app.py
   * Dockerfile
3. **Screenshots**:-- Attached in Folder.
   * Docker container running
   * API request and response for predictions.

Also all of these are available in github link- https://github.com/IYNESHDURAI/M3\_Tuning\_and\_Packaging