Machine Learning Project

Objective

In this project, you will work on building, optimizing, and deploying a machine learning model using **Azure Machine Learning**, with optional integration of **Optuna** for hyperparameter tuning, **MLflow** for experiment tracking, and **Streamlit** or **Azure Web App** for creating a front-end interface for model inference.

This end-to-end machine learning pipeline will allow you to gain hands-on experience in model training, deployment, and creating user interfaces for real-world applications.

Project Description

You will:

1. **Choose a dataset**: Select a dataset from platforms like **Kaggle**, **UCI Machine Learning Repository**, or any other public dataset of your choice. The dataset should be appropriate for a classification, regression, or clustering problem.
2. **Preprocess and Explore the Data**: Perform necessary data preprocessing such as handling missing values, encoding categorical variables, and feature scaling.
3. **Build a Machine Learning Model**: Develop an initial machine learning model using algorithms like Random Forest, XGBoost, or others, depending on the nature of the problem.
4. **Optimize the Model** (Optional):
    - Use **Optuna** for hyperparameter optimization, which can search for the best combination of hyperparameters to maximize model performance.
    - Alternatively, you can use **Azure ML's HyperDrive** if you prefer a built-in solution for hyperparameter tuning.
5. **Track Experiments with MLflow**: Use **MLflow** for tracking the training process, storing metrics, and logging model parameters.
6. **Deploy the Model**:
    - **Deploy as a REST API**: Deploy your trained model as a real-time web service using **Azure Machine Learning**.
    - **Optional Front-End**: Build an interactive front-end application to make predictions using the deployed model:
        - **Streamlit**: Create a simple Streamlit app that takes user input through a form, sends it to the Azure ML endpoint, and displays the predicted results.
        - **Azure Web App**: Alternatively, you can use an **Azure Web App** to create a web interface for model inference, allowing users to input data and receive predictions.

7. **Monitor and Evaluate**: After deployment, ensure that you monitor the model's performance, track metrics like latency and accuracy, and log results using MLflow.

Project Tasks

1. **Step 1: Setup Azure ML Workspace**
    - Create an **Azure Machine Learning workspace** in the Azure Portal.
    - Set up a **Compute Instance** for development and a **Compute Cluster** for model training (if necessary).
2. **Step 2: Data Selection and Preparation**
    - Choose a dataset from **Kaggle**, **UCI**, or another public repository.
    - Preprocess the data: handle missing values, encode categorical features, scale numerical features, and perform any necessary transformations.
3. **Step 3: Model Training and Optimization**
    - Train an initial machine learning model using the dataset.
    - Optionally, use **Optuna** to tune hyperparameters for better model performance.
    - Log your training process using **MLflow**, capturing parameters, metrics, and models.
4. **Step 4: Model Evaluation**
    - Evaluate your model using appropriate metrics such as accuracy, precision, recall, AUC, or RMSE, depending on the type of problem.
    - Fine-tune the model based on evaluation results.
5. **Step 5: Model Deployment**
    - Register your best-performing model in the **Azure ML model registry**.
    - Deploy the model as an Azure ML **real-time endpoint**.
6. **Step 6: Front-End Interface (Optional)**
    - Build a **Streamlit** or **Azure Web App** interface to allow users to input data and get predictions from the deployed model:
        - **Streamlit**: Create an app that allows users to input data, send it to the model endpoint, and display the prediction results.
        - **Azure Web App**: Alternatively, create a web app with a user-friendly form for making predictions.
7. **Step 7: Monitoring and Management**
    - Set up logging and monitoring for your deployed model (e.g., using **Application Insights**).
    - Track prediction latency and other relevant performance metrics.

Deliverables

1. **Model Pipeline**:
    - Complete model pipeline from data preparation to deployment.
    - Experiment tracking with **MLflow** and results logging.
2. **Web Interface** (Optional):

    - If using **Streamlit** or **Azure Web App**, provide a working interface that allows users to input data and receive predictions.
3. **Report/Documentation**:
    - Provide a detailed report or presentation summarizing:
        - Dataset description and preprocessing steps.
        - Model selection and training process.
        - Hyperparameter optimization (if applicable).
        - Deployment steps and web interface implementation.
        - Results, model evaluation, and any improvements made.
4. **Deployed Model**:
    - Provide access to the deployed model endpoint with example inputs and prediction results.

Additional Guidelines

- **Teamwork**: You can work individually or in teams of up to 3 members.
- **Datasets**: Choose a dataset from platforms like **Kaggle**, **UCI**, or others that fit your chosen problem type.
- **Optional Components**: Optuna and the front-end interface (Streamlit or Azure Web App) are optional but highly recommended for additional learning and project depth.