# End-to-End ML Assignment: Building a Machine Learning Pipeline

## Assignment Overview

In this assignment, you will build a complete Machine Learning pipeline from data cleaning to deployment. The assignment is divided into five main parts:

1. Data Cleaning
2. Model Building
3. Model Saving and Loading
4. FastAPI Endpoint for the Model
5. Deployment at Hugging Face with a Simple UI using Gradio or Streamlit

## Dataset

You can use any publicly available dataset that fits the requirements of the problem you are solving.

## Prerequisites

1. Python 3.7+
2. Libraries: `pandas`, `scikit-learn`, `joblib`, `fastapi`, `uvicorn`, `gradio` or `streamlit`
3. An account on Hugging Face

## Part 1: Data Cleaning

1. **Objective**: Load and clean the dataset.
2. **Steps**:
    - Load the dataset using `pandas`.
    - Handle missing values if any.
    - Encode categorical variables if any.
    - Normalize or standardize the data if necessary.

### Deliverables:

- A Python script (`data_cleaning.py`) that performs the above steps and saves the cleaned data to a CSV file.

# Part 2: Model Building

1. **Objective**: Build and evaluate a machine learning model.
2. **Steps**:
    - Split the cleaned data into training and testing sets.
    - Choose an appropriate machine learning model (e.g., logistic regression, decision tree, etc.).
    - Train the model on the training data.
    - Evaluate the model on the testing data using appropriate metrics (e.g., accuracy, precision, recall).

### Deliverables:

- A Python script (`model_building.py`) that performs the above steps and prints the evaluation metrics.

# Part 3: Model Saving and Loading

1. **Objective**: Save the trained model to a file and load it back.
2. **Steps**:
    - Use `joblib` or a similar library to save the trained model to a file.
    - Write a function to load the model from the file.

### Deliverables:

- A Python script (`model_io.py`) that contains functions to save and load the model.

# Part 4: FastAPI Endpoint for the Model

1. **Objective**: Create a FastAPI endpoint to serve the model.
2. **Steps**:
    - Create a FastAPI application.
    - Write an endpoint that accepts input data, uses the loaded model to make predictions, and returns the predictions.

### Deliverables:

- A Python script (`api.py`) that defines the FastAPI application and the prediction endpoint.

# Part 5: Deployment at Hugging Face with a Simple UI using Gradio or Streamlit

1. **Objective**: Deploy the FastAPI application and create a simple UI for it using Gradio or Streamlit.
2. **Steps**:
    - Create a Gradio or Streamlit application that interacts with the FastAPI endpoint.
    - Deploy the application on Hugging Face Spaces.

### Deliverables:

- A Gradio or Streamlit script (`app.py`) that provides a simple UI for the model.
- Instructions on how to deploy the application on Hugging Face Spaces.

# Submission Instructions

1. Create a GitHub repository for the assignment.
2. Organize the repository with separate folders for each part of the assignment.
3. Include a README file with detailed instructions on how to run each part of the assignment.
4. Submit the GitHub repository link.

# Evaluation Criteria

- Correctness and efficiency of the data cleaning process.
- Appropriate choice and evaluation of the machine learning model.
- Proper implementation of model saving and loading.
- Functionality and robustness of the FastAPI endpoint.
- Usability and deployment of the Gradio or Streamlit application.

### Bonus Points

- Use of additional machine learning techniques for better performance.
- Implementation of additional endpoints in the FastAPI application.
- Use of Docker for containerization and deployment.