# Building and Deploying a Dynamic "TruValue" ML Service

## Objective

Your task is to build a production-ready, containerized, and *dynamic* API service for our "TruValue" property valuation model. This involves not only serving predictions but also handling data preprocessing and implementing a mechanism for retraining the model with new data. This challenge is designed to test your end-to-end MLOps and software engineering skills.

## Provided Assets

You will be given a zip file containing:

1. property\_data.csv: A raw CSV file with sample property data. **Note: This data is intentionally imperfect and contains missing values and categorical features.**
2. train\_model.py: A *basic* Python script that trains a model. **You will need to enhance this script.**
3. sample\_request.json: A JSON file showing the expected raw input format for a prediction request.
4. README.md: A brief description of the project.

## Your Tasks:

1. **Enhance the Training Script (train\_model.py):**
   * Modify the script to include a robust **data preprocessing pipeline**. This pipeline must handle the missing values in the dataset and correctly encode the categorical "Location" feature before training the model.
   * The script should output two artifacts: the trained model (e.g., truvalue\_model.pkl) and the data preprocessor/encoder object (e.g., preprocessor.pkl).
2. **Build a Multi-Endpoint API Service:**
   * Create a high-performance web service in **Python (FastAPI is strongly recommended)**.
   * The service must load the trained model and preprocessor at startup.
   * Implement the following two endpoints:
     + **POST /predict**:
       - Accepts a raw JSON payload matching sample\_request.json.
       - Uses the loaded preprocessor to transform the raw input data into the format the model expects.
       - Returns a prediction from the model.
       - **Business Logic Constraint:** Implement a post-processing validation layer to ensure the API never returns a nonsensical value (e.g., a negative price).
     + **POST /retrain**:
       - This endpoint should accept a new dataset (e.g., as a file upload or JSON payload).
       - It must trigger the enhanced training script to create a new model and preprocessor.
       - Crucially, it must **safely update the in-memory model** being used by the /predict endpoint with the newly trained version **without causing any downtime or interrupting in-flight prediction requests.**
3. **Containerize the Entire System:**
   * Create a Dockerfile for your application.
   * Create a docker-compose.yml file to define and run your service. This demonstrates an understanding of multi-container environments.
4. **Provide Documentation (Update the README.md):**
   * Update the README.md with clear instructions on how to build and run the system using docker-compose.
   * Provide sample curl commands for both the /predict and /retrain endpoints.