PWSkills

High-Level Document

Industrial Automation Internship Report

Automated Machine Learning

Submitted by

Arbash Hussain

1. Purpose

The AutoML project aims to streamline the entire process of building, training, and evaluating machine learning models. It offers a web-based interface where users can upload datasets, configure training settings, evaluate model performance, and make predictions. The system supports both automatic and manual configurations for model training, making it adaptable for different levels of user expertise.

2. System Architecture

The overall architecture of the AutoML system consists of the following layers:
- Presentation Layer: Web interface built with Flask and HTML templates for user interaction.
- Service Layer: Core business logic for data handling, transformation, model training, and evaluation.
- Persistence Layer: Storage of data files, trained models, configurations, and logs.

3. Key Components

3.1 Data Ingestion

- Purpose: Load data from various sources such as uploaded files (CSV, Excel) or databases (e.g., MongoDB).
- Key Responsibilities:
  - Validate and read the input dataset.
  - Handle file storage in the specified directory (`uploads/`).
  - Save the cleaned data to `artifacts/data_ingestion/` for further processing.

3.2 Data Transformation

- Purpose: Preprocess and transform the data for model training.
- Key Responsibilities:
  - Handle missing values using different strategies (e.g., mean imputation, mode substitution).
  - Encode categorical features (e.g., one-hot encoding, label encoding).
  - Normalize numerical features.
  - Split data into training and testing sets.
  - Save transformed data to `artifacts/data_transformation/`.

3.3 Model Trainer

- Purpose: Train machine learning models based on the input data and user-specified configurations.
- Key Responsibilities:
  - Determine prediction task type (classification, regression, clustering).
  - Train models using preselected algorithms (e.g., Random Forest, Linear Regression).
  - Optimize model hyperparameters using techniques like `GridSearchCV`.
  - Save trained models to `artifacts/model_trainer/`.

3.4 Model Evaluation

- Purpose: Evaluate the performance of trained models.
- Key Responsibilities:
  - Generate predictions on the test dataset.
  - Calculate performance metrics (e.g., accuracy, precision, recall, RMSE).
  - Log metrics to a tracking system like MLflow for monitoring.
  - Display results on the web interface.

3.5 Prediction Pipeline

- Purpose: Provide predictions using the trained models on new data.
- Key Responsibilities:
  - Preprocess input data to match the format of training data.
  - Load the trained model.
  - Generate predictions and return results to the web interface.

3.6 Web Interface

- Purpose: Facilitate user interactions, including data upload, configuration, training, and prediction.
- Key Features:

- Upload datasets through a file upload form.
- Configure training settings (automatic or manual mode).
- View evaluation metrics and model logs.
- Make predictions using trained models.

3.7 Configuration and Logging

- Configuration Files:
  - `config.yaml`: Defines paths, model parameters, database configurations, etc.
  - `params.yaml`: Stores hyperparameters and other settings.
- Logging: Centralized logging for tracking activities, errors, and performance metrics in the `logs/` directory.

4. System Workflow

4.1 Data Upload

1. User uploads a dataset through the web interface.
2. The uploaded file is saved to the `uploads/` directory.

4.2 Data Preprocessing and Transformation

1. The data ingestion module reads and validates the input data.
2. Transformed data is saved to `artifacts/data_transformation/`.

4.3 Model Training

1. User selects automatic or manual configuration mode.
2. The model trainer module trains and optimizes the model using the provided data.
3. The trained model is saved to `artifacts/model_trainer/`.

4.4 Model Evaluation

1. The evaluation module loads the trained model.
2. Predictions are generated and evaluated against the test dataset.
3. Metrics are logged and displayed to the user.

4.5 Prediction

1. User provides new data through the web interface for prediction.
2. Data is preprocessed and predictions are generated using the trained model.
3. Results are displayed to the user.

5. Modes of Operation

- Automatic Mode:
  - The system automatically determines the best configurations and hyperparameters based on the dataset.
  - Suitable for users with limited ML expertise.
- Manual Mode:
  - Users manually specify configurations, such as hyperparameters and algorithms.
  - Suitable for experienced users who want greater control over the training process.

6. Technologies Used

- Web Framework: Flask
- Data Handling: Pandas, NumPy
- Model Training: Scikit-learn
- Hyperparameter Tuning: GridSearchCV
- Tracking: MLflow
- Storage: Local file storage for artifacts and configurations

7. Non-Functional Requirements

- Scalability: Designed to handle varying dataset sizes through efficient data handling.
- Performance: Optimized data processing and model training pipelines.
- User Experience: Intuitive web interface for seamless user interaction.
- Logging and Monitoring: Comprehensive logging for traceability and debugging.