PWSkills
Project Report
Industrial Automation
Internship Report

Automated Machine Learning

Submitted by
Arbash Hussain

1. Introduction

1.1 Project Overview

The AutoML system is designed to simplify the entire process of building, training, and evaluating machine learning (ML) models. It offers an intuitive web-based interface for users to upload datasets, configure training settings, evaluate models, and generate predictions. The project supports both automatic and manual modes of training, making it suitable for users with varying levels of expertise in ML.

1.2 Objectives

- Automate the end-to-end ML workflow.
- Provide flexibility with manual and automatic training configurations.
- Streamline data ingestion, preprocessing, model training, and evaluation.
- Ensure model performance can be tracked and logged.
- Offer user-friendly interaction through a web interface.

1.3 Scope

The AutoML system caters to both data science beginners and experienced practitioners. By automating complex ML tasks, it reduces time and effort, enabling users to focus on higher-level insights and decision-making.

2. System Architecture

2.1 High-Level Architecture

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

2.2 Key Modules and Components

1. Web Interface (app.py): Facilitates user interactions.
2. Data Ingestion Module: Handles loading and validation of input data.
3. Data Transformation Module: Preprocesses and transforms data.
4. Model Trainer: Trains ML models based on specified configurations.
5. Model Evaluation: Evaluates and logs model performance.
6. Prediction Module: Provides predictions on new data.
7. Logging and Configuration: Logs activities and configurations for tracking and debugging.

3. Project Structure

3.1 Directory and File Organization

- AutoML/: Core modules and utilities.
  - `data_ingestion.py`
  - `stage_02_data_transformation.py`
  - `stage_03_model_trainer.py`
  - `stage_04_model_evaluation.py`
  - pipelines/
    - `prediction_pipeline.py`
  - config/: Configuration files.
  - constants/: Constants used across the project.
  - entities/: Data structures and entities.
  - exceptions/: Custom exception handling.
  - logger/: Logging setup.
- uploads/: Directory for storing uploaded datasets.
- templates/: HTML templates for the web interface.
- logs/: Directory for log files.
- `app.py`: Main Flask application.
- `main.py`: Script for running the training pipeline.
- `requirements.txt`: Project dependencies.
- `setup.py`: Setup script for the project.

4. Functional Modules and Components

4.1 Data Ingestion

- Responsibilities:
  - Load data from files (CSV, Excel) or databases (MongoDB).
  - Validate and save data to `artifacts/data_ingestion/`.
  - Ensure data integrity (e.g., checking for null values).
- Key Functionality:
  - Reading data from user uploads.
  - Preprocessing input data for subsequent modules.

4.2 Data Transformation

- Responsibilities:
  - Preprocess data (e.g., handling missing values, normalization).
  - Encode categorical variables (e.g., one-hot encoding).
  - Split data into training and testing sets.
  - Save transformed data to `artifacts/data_transformation/`.
- Key Functionality:
  - Ensuring data is in a format suitable for model training.
  - Applying configurable preprocessing steps.

4.3 Model Trainer

- Responsibilities:
  - Determine the prediction task type (classification, regression, clustering).
  - Train models using selected algorithms (e.g., Random Forest, Linear Regression).
  - Hyperparameter tuning using `GridSearchCV`.
  - Save trained models to `artifacts/model_trainer/`.
- Key Functionality:
  - Support for automatic or manual configuration modes.
  - Selection and optimization of ML algorithms.

4.4 Model Evaluation

- Responsibilities:
  - Evaluate trained models using various metrics (e.g., accuracy, RMSE).
  - Log evaluation results to MLflow.
  - Display results on the web interface.
- Key Functionality:
  - Assessing model performance.
  - Providing evaluation reports to users.

4.5 Prediction Pipeline

- Responsibilities:
  - Preprocess new input data.
  - Load the trained model for inference.
  - Generate predictions and return results.
- Key Functionality:
  - Seamlessly integrate predictions into the user interface.

5. Modes of Operation

5.1 Automatic Mode

- Automatically configures the training pipeline based on the dataset.
- Suitable for users with limited ML expertise.

5.2 Manual Mode

- Allows users to specify configurations such as algorithms and hyperparameters.
- Suitable for experienced ML practitioners.

6. Workflow

6.1 Data Upload

1. User uploads a dataset via the web interface.
2. Dataset is saved in the `uploads/` directory.

6.2 Data Preprocessing

1. Data ingestion module reads and validates input data.
2. Data transformation module applies preprocessing steps.
3. Transformed data is saved for training.

6.3 Model Training

1. User selects automatic or manual configuration.
2. Model trainer trains and optimizes models using specified parameters.
3. Trained models are saved for evaluation.

6.4 Model Evaluation

1. Trained models are evaluated against test data.
2. Evaluation metrics are logged and displayed to users.

6.5 Prediction

1. User inputs new data for prediction.
2. Data is preprocessed, and predictions are generated using the trained model.

7. Web Interface

- Features:
  - File upload form for dataset ingestion.
  - Configuration settings for training.
  - Display of model evaluation metrics.
  - Prediction form for generating results.

8. Technologies Used

- Web Framework: Flask
- Data Processing: Pandas, NumPy
- Machine Learning: Scikit-learn
- Hyperparameter Tuning: GridSearchCV
- Tracking and Logging: MLflow, custom logger
- Storage: Local filesystem for storing datasets, models, and configurations

9. Non-Functional Requirements

- Scalability: Ability to handle different dataset sizes.
- Performance: Efficient data processing and training.
- User Experience: Easy-to-use web interface.
- Logging and Monitoring: Comprehensive logging for traceability.

10. Challenges and Solutions

- Challenge: Handling large no of features.
  - Solution: Data preprocessing optimizations and efficient memory management.
- Challenge: Providing user-friendly manual configuration options.
  - Solution: Simplified UI components for selecting configurations.

11. Future Enhancements

- Integration with cloud storage and computation resources.
- Support for additional ML frameworks (e.g., TensorFlow, PyTorch).
- Advanced visualization of model performance metrics.
- Integration of automated feature engineering techniques.

12. Conclusion

The AutoML system provides an end-to-end solution for automating machine learning workflows, empowering users to build, train, evaluate, and deploy models with ease. Its flexible architecture and user-friendly interface make it a valuable tool for both novice and expert data scientists.

5. future work

- Better algorithm for Feature Selection and extraction.
- More Customizability in Manual Config.
- More Databases support.

6. References

- https://scikit-learn.org/stable/documentation.html
- https://faker.readthedocs.io/en/master/
- https://flask.palletsprojects.com/en/2.0.x/
- https://flask-socketio.readthedocs.io/en/latest/