PWSkills
Low-Level Document
Industrial Automation Internship Report

Automated Machine Learning

Submitted by
Arbash Hussain

Project: Automated Machine Learning

Overview

This document provides a detailed low-level design for the Automated Machine Learning project. It includes detailed descriptions of modules, classes, functions, and interactions within the system.

1. Data Ingestion Module (`data_ingestion.py`)

Purpose

Responsible for loading data from various sources, such as uploaded files, databases (e.g., MongoDB), and saving it to the specified directory for further processing.

Workflow

1. **Input**: File(s) from the `uploads/` directory.
2. **Processes**:
    - Validate file format (e.g., CSV, Excel).
    - Read data using libraries such as `pandas`.
    - Copy data to `artifacts/data_ingestion/data.csv`.
    - Handle missing or invalid data during ingestion (e.g., by logging errors).
3. **Output**: `artifacts/data_ingestion/data.csv` for further processing.

2. Data Transformation Module (`stage_02_data_transformation.py`)

Purpose

Performs data preprocessing tasks to prepare the dataset for model training.

Workflow

1. **Input**: `artifacts/data_ingestion/data.csv`.
2. **Processes**:
    - Handle missing values (e.g., mean imputation for numerical features, mode for categorical).
    - Normalize numerical features (e.g., using min-max scaling or standardization).
    - Encode categorical features (e.g., one-hot encoding, label encoding).
    - Remove duplicates.
    - Split data into training and testing sets (e.g., 80%-20% split).
    - Save transformed data to `artifacts/data_transformation/`.
3. **Output**: Transformed data saved as `train_data.csv` and `test_data.csv`.

3. Model Trainer Module (`stage_03_model_trainer.py`)

Purpose

Train machine learning models based on user-defined configurations or automatically determined settings.

Workflow

1. **Input**: Transformed data from the data transformation stage (`train_data.csv`, `test_data.csv`).
2. **Processes**:
    - Identify prediction type (classification or regression) based on target variable.
    - Select appropriate models (e.g., `RandomForestClassifier`, `LinearRegression`).
    - Train models using cross-validation.
    - Optimize hyperparameters using techniques like `GridSearchCV`.
    - Save the trained model to `artifacts/model_trainer/`.
3. **Output**: Trained model file (e.g., `trained_model.pkl`).

4. Model Evaluation Module (`stage_04_model_evaluation.py`)

Purpose

Evaluate trained models using various performance metrics and log results.

Workflow

1. **Input**: Trained model and testing data (`test_data.csv`).
2. **Processes**:
    - Load the trained model.
    - Generate predictions on the test dataset.
    - Compute evaluation metrics (e.g., accuracy, precision, recall, RMSE).
    - Log evaluation metrics to MLflow.
3. **Output**: Evaluation results logged to MLflow.

5. Prediction Pipeline (`prediction_pipeline.py`)

Purpose

Make predictions using the trained model for new data provided by the user.

Workflow

1. **Input**: New data input by the user through the web interface.
2. **Processes**:
    - Load the trained model.
    - Apply necessary preprocessing to input data (same transformations as training data).
    - Generate predictions.
3. **Output**: Predicted values displayed on the web interface.

6. Web Interface (`app.py`)

Purpose

Provide a user interface for data upload, configuration, training, evaluation viewing, and prediction.

Workflow

1. **Endpoints**:
    - `/upload`: Accept file uploads.
    - `/train`: Trigger model training (automatic or manual mode).
    - `/evaluate`: Display model evaluation results.
    - `/predict`: Handle predictions for new data.
2. **Processes**:
    - Render HTML templates.
    - Handle user inputs and requests.

7. Configuration Files (`config.yaml` and `params.yaml`)

Purpose

Define paths, database settings, model parameters, etc.

Additional Components

- **Logging (`logger.py`)**: Centralized logging for all modules.
- **Error Handling (`exceptions.py`)**: Custom exceptions for error management.

Conclusion

The low-level design document provides a comprehensive breakdown of each component within the automated ml project, highlighting the purpose, methods, and interactions of various modules. This ensures clarity in understanding the system and aids in maintaining and extending the project.