Low-Level Design (LLD) for Automated Machine Learning Solution

| Written By | Mahenoor Merchant |
| --- | --- |
| Document Version | 0.1 |
| Last Revised Date | 11-November-2024 |

**Document Control**

**Change Record:**

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| 0.1 | 11-Nov-2024 | Mahenoor Merchant | Introduction & Architecture defined |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Reviews:**

| **Version** | **Date** | **Reviewer** | **Comments** |
| --- | --- | --- | --- |
|  |  |  |  |

**Approval Status:**

| **Version** | **Review**  **Date** | **Reviewed By** | **Approved By** | **Comments** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

# Introduction

## What is Low-Level design document?

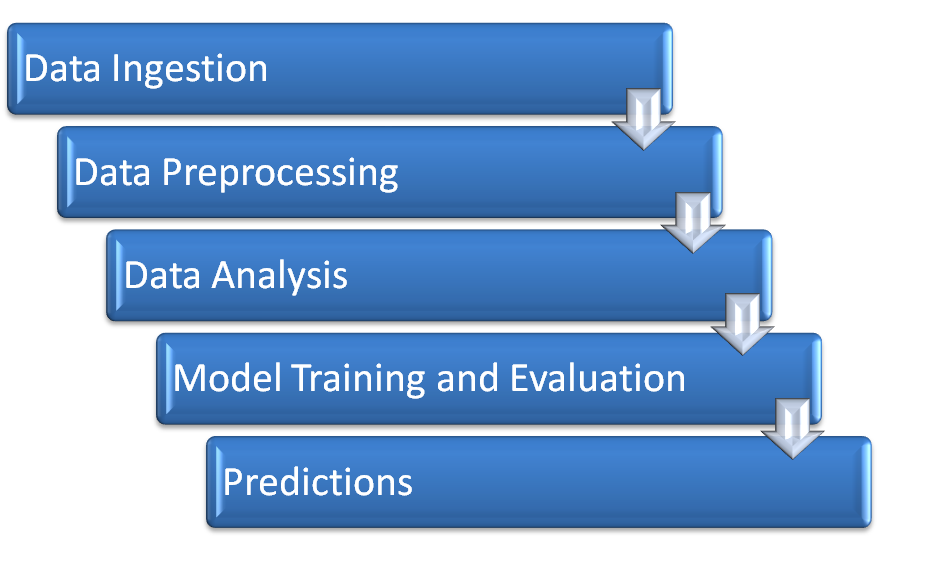
The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Automated machine learning. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

**Scope**

Low-level design (LLD) is a component-level design process that follows a step-by-

step [refinement](https://en.wikipedia.org/wiki/Refinement_(computing)) process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

Architecture



# Architecture Description

## Data Description

The user can enter data through “/predict\_classification” method if the problem type is classification problem and through “/predict\_regression” if it is a regression problem.

## Data Preprocessing

It preprocesses raw data for machine learning by separating features (X) and the target variable (y), handling missing values, scaling numerical features, encoding categorical features, and splitting the data into training, testing, and validation sets. It identifies numerical and categorical columns, applies appropriate transformations using pipelines, and saves the processed data and transformation objects for future use. This ensures the data is clean, standardized, and ready for model training.

## Data Analysis

Automates comprehensive data analysis by generating visualizations, descriptive statistics, and a detailed report in a Word document format. It ensures that both numerical and categorical data are analyzed for insightful understanding.

**Key Features of the Script:**

**1. Numerical Data Analysis**

- Creates distribution plots for all numerical features.

- Generates a correlation heatmap to identify relationships between numerical variables.

**2. Categorical Data Analysis**

- Visualizes the distribution of categorical features using count plots.

**3. Descriptive Statistics**

- Provides detailed descriptive statistics of the dataset (mean, median, standard deviation, etc.).

- Includes DataFrame information like column types, null counts, and memory usage.

**4. Word Document Report**

- Combines numerical and categorical plots, correlation heatmap, and descriptive statistics into a formatted Word document.

- Displays descriptive statistics in a table and includes DataFrame metadata for a comprehensive overview.

## Model Training and Evaluation

**Classification Model Training**

Trains multiple classification models, evaluates their performance based on accuracy, and identifies the best-performing model using validation data. The best model is then saved as a serialized file for future use. It automates the process of selecting the most accurate classification model for a given dataset.

Models Included:

1. Logistic Regression

2. Support Vector Classifier (SVC)

3. Decision Tree Classifier

4. Random Forest Classifier

5. Gradient Boosting Classifier

6. AdaBoost Classifier

7. K-Nearest Neighbors (KNN) Classifier

8. XGBoost Classifier

9. CatBoost Classifier

10. Gaussian Naive Bayes

11. Multinomial Naive Bayes

12. Bernoulli Naive Bayes

The script ensures efficient model selection for classification tasks.

**Regression Model Training**

It evaluates and selects the best-performing model from the following regression algorithms:

1. Linear Regression

2. Lasso Regression

3. Ridge Regression

4. ElasticNet Regression

5. Support Vector Regressor (SVR)

6. Decision Tree Regressor

7. Random Forest Regressor

8. Gradient Boosting Regressor

9. AdaBoost Regressor

10. XGBoost Regressor

11. K-Nearest Neighbors (KNN) Regressor

12. Bayesian Ridge Regression

**Prediction**

It automates the prediction process using a pre-trained model and test dataset. It reads the necessary data, generates predictions, and saves the output for further use.

Key Features are as follows:

1. Model and Data Loading

- Loads the pre-trained model from a serialized file (`model.pkl`).

- Reads the test dataset (`X\_test.csv`) for generating predictions.

2. Prediction Generation

- Uses the loaded model to predict outcomes for the test data.

- Combines predictions with the original test data for a comprehensive result.

3. Result Saving

- Ensures the output directory exists and saves the predictions in a CSV file (`predicted\_file.csv`).

- The output file contains both the test data and corresponding predictions for easy analysis.

The pipeline efficiently automates the final step in the machine learning process, providing accessible prediction results in a structured format.

**Model Deployment**

The model can then be deployed on AWS or GCP according to users choice through “/deploy” method which renders a dropdown from which the user can select the deployment platform.

**The analysis document, predicted files, and trained model are sent to the user via Python's send\_file() function.**