Low Level Design

Credit Card Default Detection

Written by: Abhishek Nishad

Version:07.01

Last Revised Date: 07July2023

# Contents

* Introduction.
* [What is Low-Level design document?](https://docs.google.com/document/d/11ZsH8kpfHudKxEpP882RrH1Ay4Ax44Sf/edit#heading=h.30j0zll)
* Scope
* Architecture
* Architecture Description
* Data Description
* Data Collection
* Data Conversion
* Data Insertion into Database
* Export Data from Database
* Data Preprocessing
* Feature Engineering
* Model Training
* Model Evaluation
* Model Selection
* Hyperparameter Tuning
* Model Saving
* Application Building
* Data User Input
* Data Validation
* Insertion into Database
* Data Transformation
* Model Loading
* Prediction
* Saving Output to Database
* Unit Test Cases

# Introduction

##### What is Low Level Design Document?

A low-level design (LLD) document is a technical document that describes the detailed design of a machine learning project. It refers to the detailed design phase where you define the specific components, algorithms, and technical details of the system. It involves breaking down the high-level design into smaller modules or components and determining how they will interact with each other.

##### 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

Data Requirements

Data Transformation

Start

Export From Database as csv

Data Preprocessing

Data Insertion into Database

Feature Engineering

Model Training

Get best Model on Performance

Cloud Setup

Hyperparameter Tuning Model

Model Saving

Pushing App to Cloud

Application Start

Data From User

Data Transformation

Data Insertion into DB

Data Validation

Prediction

Model Load for User Data

Export Data as CSV

End

Saving Output to DB

# Architecture Description

### Data Description

Data Consist of 30000 records and 25 features.

This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005.

### Data Collection

Data is collected from UCI Machine Learning Repository . This Data is extracted from website and in csv format.

### Data Conversion

Data is converted from csv to json format for storing in Mongo Databse.

### Data Insertion in DataBase

Mongo Database connection is created. Create a database with name passed. If the database is already created, open the connection to the database.

Collection is created in database

Insertion of records in the collection .

### Export Data From Database:

Data Export from Database - The data in a stored database is exported as a CSV file to be used for Data Pre-processing and Model Training.

### Data Preprocessing

In this step we are cleaning our data. Missing values are handled using Imputation . For Numerical variable we use standard scaler to scale the values in same range. For categorical values we convert them into One hot Encoding representation. The preprocessor pickle file is then saved

### Feature Engineering

We are going to analyise the data for any outliers, Feature extraction , Feature Selectionor transforming the input features to improve the performance of a machine learning model.

### Model Training

After feature Engineering we split our data for training and validation. Training is done using multiple ML algorithms and performance is checked for very ML model. The Best Model is selected with highest performance.

### Hyperparameter Tuning

After Selecting the best Model we hyperparameter tune it to select the best parameter fitting our requirements.

### Model Saving

The model is saved so that it can be loaded for unseen data and make appropriate prediction

### Cloud Setup

Cloud setup is done so that it can be scalable, Cost Efficient ,Flexibility ,Agility and Security. There are several cloud service providers to choose from, such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), or IBM Cloud. WE have choosen AWS cloud for this proect.

### Pushing App to Cloud

 Once your app is packaged, you can deploy it to the cloud. This process will vary depending on the cloud computing platform you are using.

### Data From User

User inputs the data on app and this data is further transferred for prediction

### Data Validation

User Data is validated to check for proper inputs if not will throw an error to user to enter valid data

### Data Transformation

Data entered by user is transformed in appropriate format to be pushed in Database.Here our preprocessor file is loaded for preprocessing data.

### Data Insertion Into DB

The data Collected from user is Stored in Database. Here we use Mongo DB which stores in json format

### Export Data as CSV

The data is then exported into csv format so that it can be utilised by model

### Model Load

Model is loaded and user data is fed to this model

### Prediction

Model will give prediction on user data whether the data entered can have impact for defaut on loan. That is if person can default on repayment or not.

### Saving Output to DB

The output is further saved in Mongo DB.

# Unit Test Cases

|  |  |  |
| --- | --- | --- |
| Test Case Description | Pre-Requisite | Expected Result |
| Verify whether the Application URL is  accessible to the user | 1. Application URL  should be defined | Application URL should be  accessible to the user |
| Verify whether the Application loads completely for the user when the URL is accessed | 1. Application URL is accessible 2. Application is deployed | The Application should load completely for the user when the URL is accessed |
| Verify whether user is able to see input fields on logging in | 1. Application is accessible 2. User is signed up to the application 3. User is logged in   to the application | User should be able to see input fields on logging in |
| Verify whether user is able to edit all input fields | 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application | User should be able to edit all input fields |
| Verify whether user gets Submit button to submit the inputs | 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the applicatio | User should get Submit button to submit the inputs |
| Verify whether user is presented with recommended results on clicking  submit | 1. Application is accessible 2. User is signed up to the application 3. User is logged in   to the application | User should be presented with recommended results on clicking  submit |
| Verify whether the recommended results are in accordance to the selections user made | 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application | The recommended results should be in accordance to the selections user made |
| Verify whether KPIs modify as per the user inputs for the user's health | 1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application | KPIs should modify as per the user inputs for the user's health |
|  |  |  |