**Low Level Design**

**Credit Card Defaulters Prediction System**

|  |  |
| --- | --- |
| Written By | Subhajit Roy |
| Document Version | 1 |
| Last Revision Date | 06-Mar-2022 |

Document Control

Change Record:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 1 | 06-Mar-2022 | Subhajit Roy |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Reviews:

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

Approval Status:

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

Contents

[1. Introduction 1](#_Toc97417912)

[3.1 What is Low-Level design document? 1](#_Toc97417913)

[3.2 Scope 1](#_Toc97417914)

[2. Architecture 2](#_Toc97417915)

[3. Architecture Description 3](#_Toc97417916)

[3.1 Data Description 3](#_Toc97417917)

[3.2 Data Transformation & Validation 3](#_Toc97417918)

[3.3 Existing Database Table Deletion 3](#_Toc97417919)

[3.4 Data Insertion into Database 4](#_Toc97417920)

[3.5 Export Data from Database 4](#_Toc97417921)

[3.6 Data Pre-processing 4](#_Toc97417922)

[3.7 Data Clustering 4](#_Toc97417923)

[3.8 Model Building 4](#_Toc97417924)

[3.9 Data from User 4](#_Toc97417925)

[3.10 Data Validation 4](#_Toc97417926)

[3.11 Data Clustering 4](#_Toc97417927)

[3.12 Model Call for Specific Cluster 4](#_Toc97417928)

[3.13 Defaulter prediction & saving output in folder 4](#_Toc97417929)

[3.14 Deployment 4](#_Toc97417930)

[4. Unit Test Cases 5](#_Toc97417931)

# Introduction

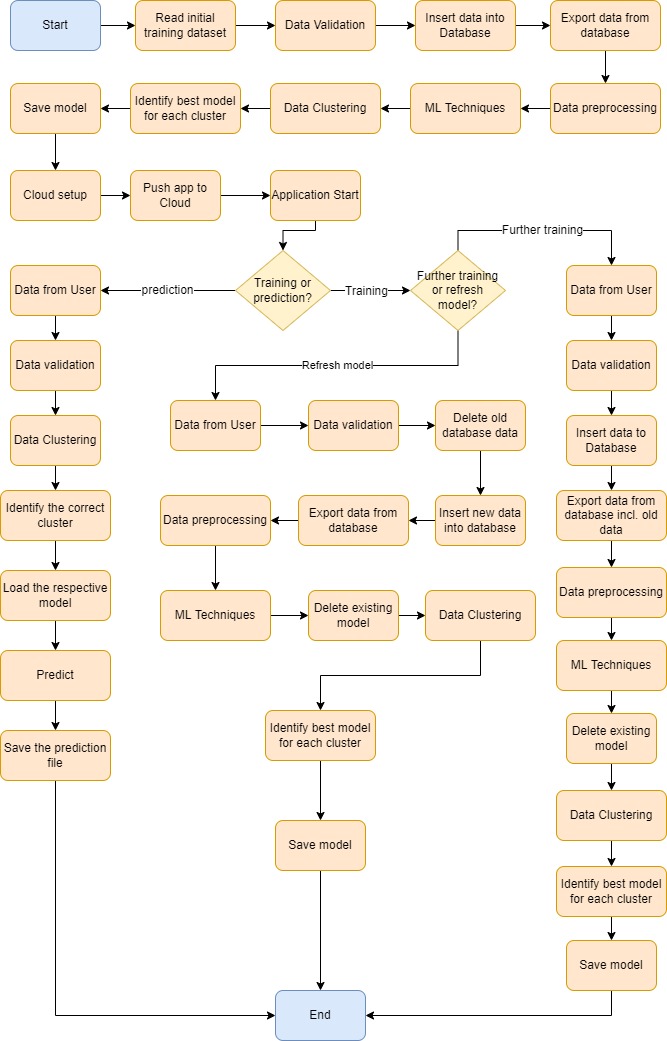
## What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation System. 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 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

This dataset is a CSV file that 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. There are 25 variables.

|  |  |
| --- | --- |
| **Feature Name** | **Feature Description** |
| ID | ID of each client |
| LIMIT\_BAL | Amount of given credit in NT dollars (includes individual and family/supplementary credit |
| SEX | Gender (1=male, 2=female) |
| EDUCATION | (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown) |
| MARRIAGE | Marital status (1=married, 2=single, 3=others) |
| AGE | Age in years |
| ***Repayment Status:*** *(-1=pay duly, 1=payment delay for one month, 2=payment delay for two months…8=payment delay for eight months, 9=payment delay for nine months and above)* | |
| PAY\_0 | Repayment status in September, 2005 |
| PAY\_2 | Repayment status in August, 2005 |
| PAY\_3 | Repayment status in July, 2005 |
| PAY\_4 | Repayment status in June, 2005 |
| PAY\_5 | Repayment status in May, 2005 |
| PAY\_6 | Repayment status in April, 2005 |
| BILL\_AMT1 | Amount of bill statement in September, 2005 (NT dollar) |
| BILL\_AMT2 | Amount of bill statement in August, 2005 (NT dollar) |
| BILL\_AMT3 | Amount of bill statement in July, 2005 (NT dollar) |
| BILL\_AMT4 | Amount of bill statement in June, 2005 (NT dollar) |
| BILL\_AMT5 | Amount of bill statement in May, 2005 (NT dollar) |
| BILL\_AMT6 | Amount of bill statement in April, 2005 (NT dollar) |
| PAY\_AMT1 | Amount of previous payment in September, 2005 (NT dollar) |
| PAY\_AMT2 | Amount of previous payment in August, 2005 (NT dollar) |
| PAY\_AMT3 | Amount of previous payment in July, 2005 (NT dollar) |
| PAY\_AMT4 | Amount of previous payment in June, 2005 (NT dollar) |
| PAY\_AMT5 | Amount of previous payment in May, 2005 (NT dollar) |
| PAY\_AMT6 | Amount of previous payment in April, 2005 (NT dollar) |
| default.payment.next.month | Default payment (1=yes, 0=no) |

## Data Transformation & Validation

The data is read from the CSV file and validated so that it can be inserted into the database. The validation includes checks like proper number of columns, marking the missing data as NULL etc.

## Existing Database Table Deletion

If the user uploads new training data and selects the option to delete the old data from the front end of the application, then the existing database table is deleted along with the existing data and new database table is created. The new database table is not created, if the user wishes to further train the model using new dataset.

## Data Insertion into Database

Connection is made with the database and the data is inserted into the database table.

## Export Data from Database

The data from the database is collected and stored in the CSV file to be used for data processing and model training.

## Data Pre-processing

The steps for data pre-processing includes imputation of the null values, imbalanced dataset handling, scaling of the numerical fields, etc.

## Data Clustering

Clustering is made on the training data. The model is saved. The optimal number of clusters is determined. A new column is inserted into the dataset to mark the cluster number in which each record belongs.

## Model Building

For each cluster, several models were built and trained. Their scores were compared and the best model is chosen for that particular cluster. The model is saved along with the cluster number to identify the model that needs to be used for predictions.

## Data from User

The data that needs to be predicted is uploaded by the user from the front end.

## Data Validation

The data is read from the CSV file and validated.

## Data Clustering

The clustering model is loaded and the predicted data is being used to identify the cluster that each record belongs to.

## Model Call for Specific Cluster

The records are grouped by clusters and the corresponding best model is loaded and the data is passed to it to get the prediction.

## Defaulter prediction & saving output in folder

The predicted output is saved in a CSV file in the S3 bucket.

## Deployment

The model is deployed on Amazon AWS.

# 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 the user is able to upload the file for training | 1. Application is accessible  2. The browse button of the application is working properly | The user should be able to upload the file |
| Verify whether the model has been trained or not | 1. Application is accessible  2. The browse button of the application is working properly | The user should be able to see a success message at the end |
| Verify whether the user is able to upload the file for prediction | 1. Application is accessible  2. The browse button of the application is working properly | The user should be able to upload the file |
| Verify whether the prediction has been made or not | 1. Application is accessible  2. The browse button of the application is working properly | The user should be able to see a success message at the end |