Low Level Design (HLD)

Credit Card Default Prediction

Revision Number: 1.0

Last date of revision: 25/09/2023

# 

# Document Version Control

**Change Record**

| Date | Version | Author | Comments |
| --- | --- | --- | --- |
| 25/09/2023 | v1.0 | Abraham Audu | Introduction, Architecture and Architecture Description |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Reviews**

| Date | Version | Reviewer | Comments |
| --- | --- | --- | --- |
|  | v1.0 |  |  |
|  |  |  |  |

**Approval Status**

| Review Date | Version | Reviewed By | Approved By | Comments |
| --- | --- | --- | --- | --- |
|  | v1.0 |  |  |  |
|  |  |  |  |  |

**Table of Contents**

[Document Version Control 2](#_heading=h.gjdgxs)

[**Abstract 4**](#_heading=)

[**1. Introduction 5**](#_heading=)

[1.1 Why this High-Level Design Document? 5](#_heading=)

[1.2 Scope 5](#_heading=h.d1qeboog8101)

[**2. General Description 5**](#_heading=h.ggkmsajlo83h)

[2.1 Product Perspective 5](#_heading=h.47rs17phkyud)

[2.2 Problem Statement 5](#_heading=h.2m4jh7xphb00)

[2.3 Proposed Solution 5](#_heading=h.a24luzxqjj4y)

[2.4 Further Improvements 6](#_heading=h.9tipoxtfyl22)

[2.5 Technical Requirement 6](#_heading=h.6e6nbndd0z0)

[2.6 Data Requirements 6](#_heading=h.u8e5flsgmxuv)

[2.7 Tools Used 6](#_heading=h.eeubfnt9ay1j)

[2.8 Constraints 6](#_heading=h.uq8kg0k4hxoi)

[2.9 Assumptions 6](#_heading=h.jy1d8hqituqs)

[**3. Design Details 7**](#_heading=h.4o154eianxnu)

[3.1a Development Process Flow 7](#_heading=h.2izcvvqfv1k2)

[3.1b Deployment Process Flow 7](#_heading=h.gegq5e7jn42o)

[3.2 Event Log 7](#_heading=h.7vujpp5x3o0f)

[3.3 Error Handling 8](#_heading=h.754hode35qq)

[**4. Performance 8**](#_heading=h.83j129w5xneh)

[4.1 Reusability 8](#_heading=h.k9trcoggc87)

[4.2 Application Compatibility 8](#_heading=h.va0xe28dsx4t)

[4.3 Resource Utilisation 8](#_heading=h.yci7e8mbhjs5)

[4.4 Deployment 8](#_heading=h.rwu1iljkevt8)

[**5. Conclusion 8**](#_heading=h.j1a2hkradlcn)

# Introduction

## Why this Low-Level Design Document?

The purpose of this Low-Level Design Document (LLD) is to give the internal logical design of the program code for the Credit Card Default Prediction Solution. This LLD describes the modules, functions and relationships between them. It describes the modules in a way that gives the programmer enough information to write said modules and functions only by reading this document.

## 1.2 Scope

This LLD is a component level design process that follows a step-by-step process to determine and refine what happens at each step of the entire process. This process factors the data structures used, software architecture, the flow of the source code and performance algorithms. Data organisation may be determined during requirement analysis,but there is still room for refinement during the data design work.

# 2. Architecture



# 3. Architecture Description

## 3.1 Data Description

The data is a CSV file containing 30,000 rows of data with each row representing a customer. The data consists of personal information like age, gender, level of education and marital status. In addition the data has financial information covering punctuality of payments, credit card bills and corresponding credit card bill payments for the last six months.

## 3.2 Data Pre-Processing

In the pre-processing phase, the “ID” column is dropped, features with less than 20 unique values are one-hot encoded and targets which are non-defaulting users are undersampled to solve for class imbalance in the data. After splitting the data, the features are scaled with sk-learn’s standard scaler based on a fit to the training set.

## 3.3 Model Training and Evaluation

After preprocessing the data, Logistic Regression, Support Vector Machine, Multi-Layer Perceptron Classifier and Random Forest models are trained and tested with best parameters based on GridSearchCV from model experimentation phase. The models are evaluated based on the weighted f1 score of both target classes.

## 3.4 Data from User

The user will upload a CSV file containing the records of the credit card owner being evaluated via the user interface of the Credit Card Default Prediction app.

## 3.5 Data Validation

The data supplied by the user will be validated for completeness and data types for the individual features required by the model.

## 3.6 Inference Generation

After data validation, the user will then be allowed to choose the model they want to use to determine the probability that the credit card owner will default. When the user clicks “Predict”, the data is sent as a request to the server where the appropriate model will be loaded to memory and inference will be generated and sent back as response. This response will then be parsed and displayed to the user in an organised format.

## 3.7 Deployment

The Application will be deployed locally.

### 3.7.1 Local Deployment Details

Code Repository: <https://github.com/theabrahamaudu/credit_card_default_predictor>

Steps:

* Clone repository
* Create virtual environment
* Open terminal
* Install requirements: python -m pip install -r requirements.txt
* Start FastAPI server: python -m src/main
* Start Streamlit frontend: streamlit run src/streamlit\_frontend.py

Refer to README.md on the code base for more details.

# 4. Unit Test Cases

| **Test Case Description** | **Prerequisite** | **Expected Result** |
| --- | --- | --- |
| Verify whether undersampling function works with binary targets | target column is encoded | Oversampled targets are randomly removed |
| Verify whether undersampling function works with single value targets | target column is encoded | Number of target rows stays intact |
| Good data uploaded by user is positively validated | Data uploaded by user | validator passes |
| Data with unrecognised feature is uploaded by user | Data uploaded by user | Validator flags unrecognised feature |
| Data with incomplete set of features uploaded by user | Data uploaded by user | Validator flags unprovided feature as missing |
| Data with invalid data type uploaded by user | Data uploaded by user | Validator flags features with incorrect data type |