

**Credit Card Defaulter Predictor App**

High Level Design

Domain: Machine Learning

Creator: Sudhanshu Rastogi

Date: 04.09.2023

Github: https://github.com/sudhanshu2198

**Document Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Date issued | Version | Description | Author |
| August 28, 2023 | 1.1 | First Draft | Sudhanshu Rastogi |
| September 3, 2023 | 1.2 | Added documents files | Sudhanshu Rastogi |
| September 3, 2023 | 1.3 | Correcting font sizes | Sudhanshu Rastogi |

Contents

[Abstract 3](#_Toc110595993)

[Introduction 3](#_Toc110595994)

[What is High-Level Design Document? 3](#_Toc110595995)

[Scope 4](#_Toc110595996)

[General Description 4](#_Toc110595997)

[Definitions 4](#_Toc110595998)

[Product Description 4](#_Toc110595999)

[Problem Statement 4](#_Toc110596000)

[Proposed solution 5](#_Toc110596001)

[Further improvements 5](#_Toc110596002)

[Data requirements 5](#_Toc110596003)

[Tools used 5](#_Toc110596004)

[Hardware Requirements 7](#_Toc110596005)

[Constraints 7](#_Toc110596006)

[Assumptions 7](#_Toc110596007)

[Design Details 7](#_Toc110596008)

[Process Flow 7](#_Toc110596009)

[Event Log 8](#_Toc110596010)

[Error Handling 9](#_Toc110596011)

[Performance 9](#_Toc110596012)

[Reusability 9](#_Toc110596013)

[Application Compatibility 9](#_Toc110596014)

[Resource Utilization 9](#_Toc110596015)

[Dashboards 10](#_Toc110596016)

[KPIs (Key Performance Indicators) 10](#_Toc110596017)

[Conclusion 10](#_Toc110596018)

# Abstract

Credit Card payment default occurs when you fail to pay the Minimum Amount Due

(MAD) on the credit card for a few consecutive months. Usually, the default notice is

sent by the card issuer after 6 consecutive missed payments.

**Consequences of Credit card payment default**

Lawful Punishments Suspended Credit Card Account Detrimental Effect on Credit Score

High-Interest Rates Asset Possession In this project we classify customers as potential

defaulter given personal and 6 months banking details.

**The main goal of this project is to develop a Credit Card Defaulter Prediction**

**model and deploying the model using streamlit as an Web App.**

# Introduction

## What is High-Level Design Document?

The goal of this HLD or a high-level design document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

* Present all of design aspects and define them in detail
* Describe all user interfaces being implemented
* Describe the hardware and software interfaces
* Describe the performance requirements
* Include design features and architecture of the project
* List and describe the non-functional attributes such as security, reliability, maintainability, portability, reusability, application compatibility. resource utilization, serviceability

## Scope

The HLD documentation presents the structure of the system, such as database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly technical terms which should be understandable to the administrators of the system.

# General Description

## Definitions

|  |  |
| --- | --- |
| Term | Description |
| Credit Card Defaulter Prediction App | CCDPA |
| Database | Collection of the Information |
| Cloud | A data center full of services connected to the internet performing service |
| IDE | Integrated Development Environment |
| UI | User Interface |
| Streamlit | Open-source framework used for model deployment |
| Github | platform for hosting code that allows for version control and collaboration |

## Product Description

Credit Card Defaulter Predictor is a Machine Learning based classification model which helps us to do predictive analysis of the credit card defaulting certain features and parameters.

## Problem Statement

To create an ML based solution for predictive modelling of credit card defaulting and also deploy it as an interactive web app.

## **The main goal of this article is to deploy trained machine learning model predicting whether a customer will default on credit card payments as an web app using streamlit, visualizing features variable, bivariate analysis and model experimenting results using matplotlib library and insights on about the factors affecting the defaulting of credit card due payments.**

## Proposed solution

Using all the standard techniques used in the life cycle of a Data Science project starting from Data Exploration, Data Cleaning, Feature Engineering, Model Selection, Model Building and Model Testing and also building a frontend where a user can fill their information in the form input and get the output instantly.

## Further improvements

This Credit Card Defaulter Prediction model can be easily embedded inside any website or an application and everybody can get quick answer by inputting required data on friendly user interface.

This can be further improved by training more data in the model. Data can be acquired by web scrapping using scrapy or beautifulsoup library.

## Data requirements

Data requirement completely depend on our problem statement. Required dataset should contain the following features:

**Variables**

* **Credit Limit**: Amount of the given credit (in dollars): it includes both the individual
* consumer credit and his/her family (supplementary) credit
* **Sex** (1=male; 2=female)
* **Education** (1=graduate school; 2=university; 3=high school; 4=other)
* **Marital Status** (1=married; 2=single; 3=others)
* **Age** (years)
* **History of past payment**: The measurement scale for the repayment status is: -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
* **Amount of bill statement** (dollars) for past 6 months
* **Amount of previous payment** for the past 6 months

These are the required parameters to feed into model.

## Tools used

Python programming language and frameworks such as NumPy, Pandas, Scikit-learn, Plotly, LightGBM, Streamlit, and a few other libraries were used to build the whole model.





 

* For visualization tasks, matplotlib, seaborn and plotly were used.
* Streamlit was used for building the web application.
* GitHub is used as version control system
* NumPy and Pandas were used to clean and interpret data
* Scikit-learn was used to cross validate and compare different models
* Light Gradient Boosting Regressor was used to build the final model

## Hardware Requirements

* Windows Server, Linux, or any operating system that can run as a webserver, capable of delivering HTML5 content.
* Minimum 1.10 GHz processor or equivalent.
* Between 1-2 GB of free storage
* Minimum 512 MB of RAM
* 3 GB of hard-disk space

## Constraints

The front-end must be user friendly and should not need any one to have any prior knowledge in order to use it.

## Assumptions

The main objective of this project is to implement the use case as previously mentioned (2.3 problem statement) for new dataset that comes through the UI. It is assumed that all aspects of this project have the ability to work together as the designer is expecting and also the data on which our model is trained is as correct as possible

# Design Details

## Process Flow

For accomplishment of the task, we will use a trained Machine Learning model. The process flow diagram is shown below:

**Data Preparation**

**Model**

**Development**

**Deployment**

**Deployment**

## Event Log

The system should log every event so that the user will know what process is running internally. Initial step-by-step description:

1. The system identifies at what level logging is required
2. The system should be able to log each and every system flow
3. Developer can choose logging method. You can choose database logging/ File logging as well
4. System should not hang even after so many loggings. Logging just because we can easily debug issues, so logging is mandatory to do.

## Error Handling

Errors should be encountered, an explanation will be displayed as to what went wrong ? An error will be defined as anything that falls outside the normal intended usage.

# Performance

Credit Card Defaulter Predictor model achieved a very high recall value of 0.85 for the dataset, thereby minimizing the cost for credit card issuing companies for lowering cards issuance to risky customers. Also, model retraining is very important to further enhance its performance.

## Reusability

The code written and the components used should have the ability to be reused with no problems.

## Application Compatibility

The different components for this project will be using Python as an interface between them, each component will have its own task to perform, and it is the job of Python to ensure proper transfer of information.

# Dashboards

As and when, the system starts to capture the historic/ periodic data for a user, the dashboards will be included display charts over time with progress on various indicators or factors.



## KPIs (Key Performance Indicators)

* Key Performance Indicators of MITVP
* Latency or the amount of time the application takes to display results for some specific input.
* The processing power our application takes to run
* The memory and RAM our application takes to run on a web server.

# Conclusion

All in all, overall project architecture, design details, used technologies and performance were explained in detail. The Credit Card Defaulter prediction App will give defaulting probability instantly and has the potential to help companies reduce losses by not issuing credit card to risky.