iNeuron.ai

Credit Card Default Prediction

High Level Design (HLD) Documentation

Himanshu Banodha

May 30, 2024

Document Version Control

Date Issued	Version	Description	Author
30/05/2024	1	Initial HLD – V1.0	Himanshu Banodha

Table of Contents

1. Introduction

1.1. Why this High-Level Design Document?

This purpose of this High-Level Design (HLD) 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 the design aspect and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include design features and the architecture of the project

1.2. Scope

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

1.3. Definitions

Term	***Description***
Database	Collection of all the information monitored by this system
IDE	Integrated-Development Environment

2. General Description

2.1. Product Perspective

The Credit Card Default Prediction using classification-based Machine Learning Algorithms.

2.2. Problem Statement

Financial threats are displaying a trend about the credit risk of commercial banks as the incredible improvement in the financial industry has arisen. In this way, one of the biggest threats faces by commercial banks is the risk prediction of credit clients. The goal is to predict the probability of credit default based on credit card owner's characteristics and payment history.

2.3. PROPOSED SOLUTION

The solution is a classification-based machine learning model. This can be implemented by using the different classification algorithms (such as Logistic, Random Forest, Decision Tree etc.)

To build this model we will perform Data Preprocessing which contains feature engineering, feature selection, feature transformation etc.

2.4. Technical Requirements

This document addresses the requirements for detecting who is defaulting the bill payments.

Here are some requirements for this project.

- Model should be exposed through API or User interface, so that anyone can test the model
- Model should be deployed on cloud (Azure, AWS, GCP, Heroku).

2.5. Data Requirements

Data requirement completely depends on our problem statement.

Here we got the dataset from Kaggle (Credit Card Default Prediction Dataset) this dataset contains information about default payments, demographic factors, credit data, history of payment, & bill statements of credit card clients in Taiwan from April 2005 to September 2005.

There are 25 variables:

- **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
- **PAY_0**: Repayment status in September, 2005 (-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_2**: Repayment status in August, 2005 (scale same as above)
- **PAY_3**: Repayment status in July, 2005 (scale same as above)
- **PAY_4**: Repayment status in June, 2005 (scale same as above)
- **PAY_5**: Repayment status in May, 2005 (scale same as above)
- **PAY_6**: Repayment status in April, 2005 (scale same as above)
- **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)

2.6. Tools used

Python programming language and frameworks such as NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn & Flask are used to build the whole model.

- VS Code is used as an IDE.
- For visualization of the plots Matplotlib & Seaborn are used.
- Front end development is done HTML/CSS.
- Flask is used for backend development.
- GitHub is used as a version control system.

2.7. Constraints

The Credit Card Default Prediction System must be user friendly, error free & users should not be required to know any of the back-end workings.

2.8. Assumptions

The main objective of the project is to implement the use cases as previously mentioned (2.2 Problem Statement) for new dataset that comes through any credit card holder. Machine learning based models are used for predicting defaulters in such use cases based on input data.

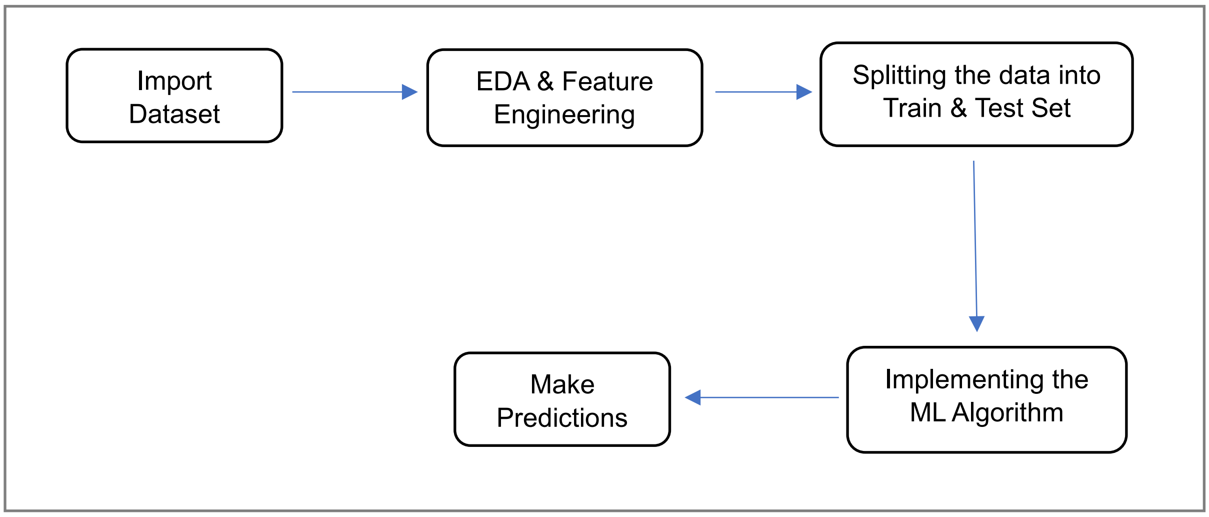
It is also assumed that all aspects of this project have the ability to work together in the way the designer is expecting.

3. Design Details

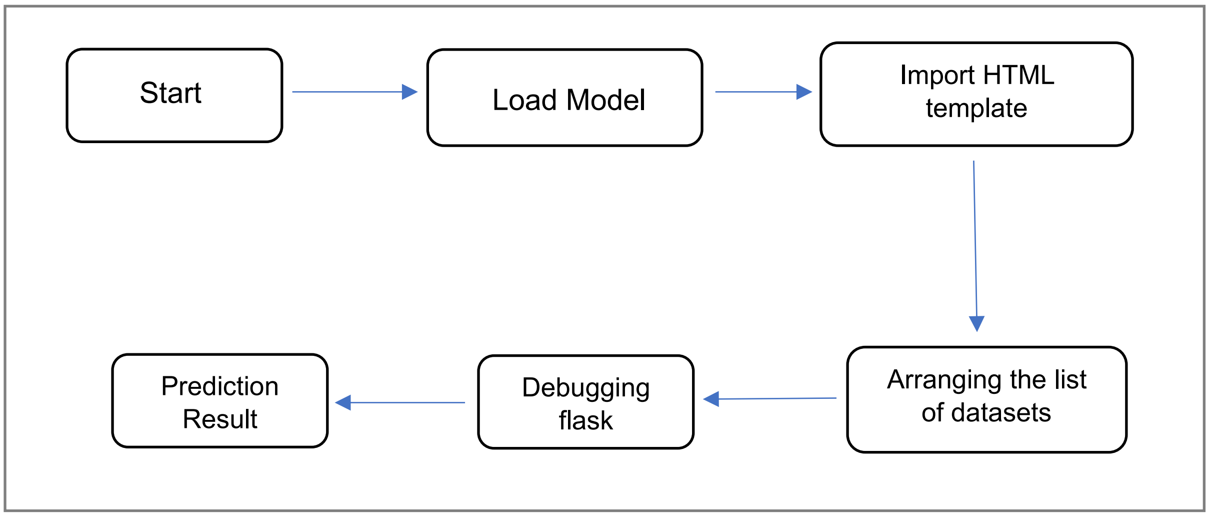
3.1. Process Flow

For identifying the defaulters, we will use a machine learning base model. Below is the process flow diagram as shown below.

Proposed Methodology:


Import Dataset
EDA & Feature Engineering
Splitting the data into Train & Test Set
Implementing the ML Algorithm
Make Predictions

3.1.1. Deployment Process



3.2. Evant 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 step logging required
2. The system should be able to log each and every system flow
3. Developers can choose logging methods. You can choose database logging/ File logging as well.
4. System should not hang even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

3.3. Error Handling
Should errors be encountered, an explanation will be displayed as to what went wrong? An error will be defined as anything that falls outside the normal and intended usage.

3.4. Performance
The credit card default prediction model is used for predicting which customers can be a defaulter in the upcoming month, so it should be as accurate as possible, so that it should as many accurate predictions as possible.

Also, model retraining is very important to improve the performance.

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

3.6. 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 the Python to ensure proper transfer of information.

3.7. Resource Utilization
When any task is performed, it will likely use all the processing power available until that function is finished.

3.8. Deployment
*Localhost

4. Conclusion
The Credit Card Default Prediction Project is designed in Flask; hence it is accessible to everyone. The above designing process will help banks & loan lenders predict whether customers will default on credit card payments or not, so the bank or respective departments can take necessary action, based on the model’s predictions. The UI is made to be user-friendly to that the user will not need much knowledge of any tools but will just need the information for results.