# High Level Design (HLD)

## Credit Card Default Detection Application

by

Abhijit Paul

Gouthami K

Revision Number: 1.0
Last date of revision: 13/12/2023

# Document Version Control

| Date Issued | Version | Description | Author |
|---|---|---|---|
| 13/12/2023 | 1 | Initial HLD - V1.0 | Abhijit Paul |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Contents

# Abstract

The Credit Card Default Detection project aims to leverage machine learning algorithms to predict the likelihood of credit card defaults based on historical data. This High-Level Design (HLD) document provides a comprehensive overview of the project's architecture, design decisions, and key features.

The project begins with data collection from diverse sources, including credit history, demographics, and transaction data. Rigorous data cleaning and preprocessing steps are implemented to handle missing values, outliers, and ensure data quality. The ML model selection involves utilizing classification algorithms such as Random Forest, Logistic Regression, and Gradient Boosting. Feature selection techniques and cross-validation are employed to enhance model performance and robustness.

Model evaluation metrics, including accuracy, precision, recall, and F1-score, are defined to measure the model's effectiveness. Cross-validation further ensures the stability of the models. The deployment phase involves deploying the model on a scalable cloud platform, with APIs exposed for seamless integration with other systems.

To ensure ongoing reliability, the project incorporates a monitoring system with defined metrics and an alerting mechanism. A re-training strategy is implemented to adapt the model to changing data patterns. Security and compliance measures are considered to protect sensitive financial data, and model explainability techniques are employed for transparency.

Optionally, a user interface or dashboard is designed for end-users to interact with predictions, enhancing user experience. Scalability planning addresses potential increases in data volume or user requests.

This HLD document serves as a guide for the development, deployment, and maintenance of the Credit Card Default Detection ML Project, emphasizing robustness, interpretability, and scalability in its design.

# 1 Introduction

## 1.1 Why this High-Level Design Document?

The 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 contradiction 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 expects and define them in detail
- Describe the user interface being implemented
- Describe the hardware and software interfaces
- Describe the performance requirements
- Include designs which are the architecture of the project
- List and describe the non functional attributes like:
    - ☐ Security
    - ☐ Reliability
    - ☐ Maintainability
    - ☐ Portability
    - ☐ Reusability

## 1.2 Scope

The HLD documentation presents structure of the system, such as database architecture, application architecture (layers), application flow (Navigation), and technology architecture.

## 1.3 Definitions

**HTML** - HTML is the standard markup language for creating Web pages
**CSS** - CSS is a style sheet language used for describing the presentation of HTML
**Flask** - Flask is a micro web framework written in Python
**Model** - Machine Learning model

# 2 Project Overview

## 2.1 Project Objective

The goal of the project is to predict credit card defaults based on historical data

## 2.2 Key Features

- Classification of credit card holders into default and non-default categories.
- Model interpretability for better decision-making.
- Integration with a user interface for ease of use.

# 3 Data Collection and Preprocessing

## 3.1 Data Source

Credit history, demographics, and transaction data.

## 3.2 Data Cleaning and Preprocessing

- Handle missing values, outliers, and data imbalances.
- Feature scaling and encoding categorical variables.

# 4 Machine Learning Model

## 4.1 Algorithm Selection

Choose classification algorithms suitable for the task
(e.g Random Forest, Logistic Regression, Gradient Boosting, KNearest Neighbours, etc).

### 4.2 Feature Selection

Identify relevant features using techniques such as feature importance.

### 4.2 Model Training

- Train multiple models and evaluate their performance.
- Use cross-validation and hyper parameter tuning  to ensure robustness.

# 5 Model Evaluation

### 5.1 Evaluation Metrics

- Define metrics: accuracy, precision, recall, F1-score.
- Explore ROC-AUC for model performance.

### 5.2 Cross Validation

Employ k-fold cross-validation to assess model stability.

# 6 Model Deployment

### 6.1 Deployment Environment

Deploy the model on a cloud platform (e.g., AWS, Azure) for scalability.

### 6.2 API Integration

Expose model predictions through APIs for integration with other systems.

# 7 Monitoring and Maintenance

## 7.1 Monitoring Metrics

- Define metrics for model performance monitoring.
- Implement an alerting system for potential issues.

## 7.2 Re-Training Strategy

Plan periodic retraining to adapt to changing data patterns.

# 8 Security and Compliance

## 8.1 Data Privacy

Implement measures to ensure the privacy of sensitive financial data.

## 8.2 Model Explainability

Utilize techniques for model explainability and interpretability.

# 9 User Interface (Optional)

## 9.1 Dashboard

Design a user interface or dashboard for end-users to interact with predictions.

# 10 Scalability

## 10.1 Scalability Plan

Plan for scaling the system to handle increased data volume or user requests.

# 11 Documentation

## 11.1 Code Documentation

Document the machine learning codebase thoroughly.

## 11.2 User Guide

Create a user guide for end-users or developers interacting with the system.

# 12 Dependencies

## 12.1 Software Dependencies

List external libraries, frameworks, and tools used in the project.

# 13 Testing

## 13.1 Unit Testing

Specify unit tests for individual components.

### 13.2 Integration Testing

Plan for integration tests to ensure seamless collaboration among different modules.

# 14 Cost Estimates

### 14.1 Infrastructure Costs

Estimate costs associated with infrastructure, cloud services, and third-party tools.

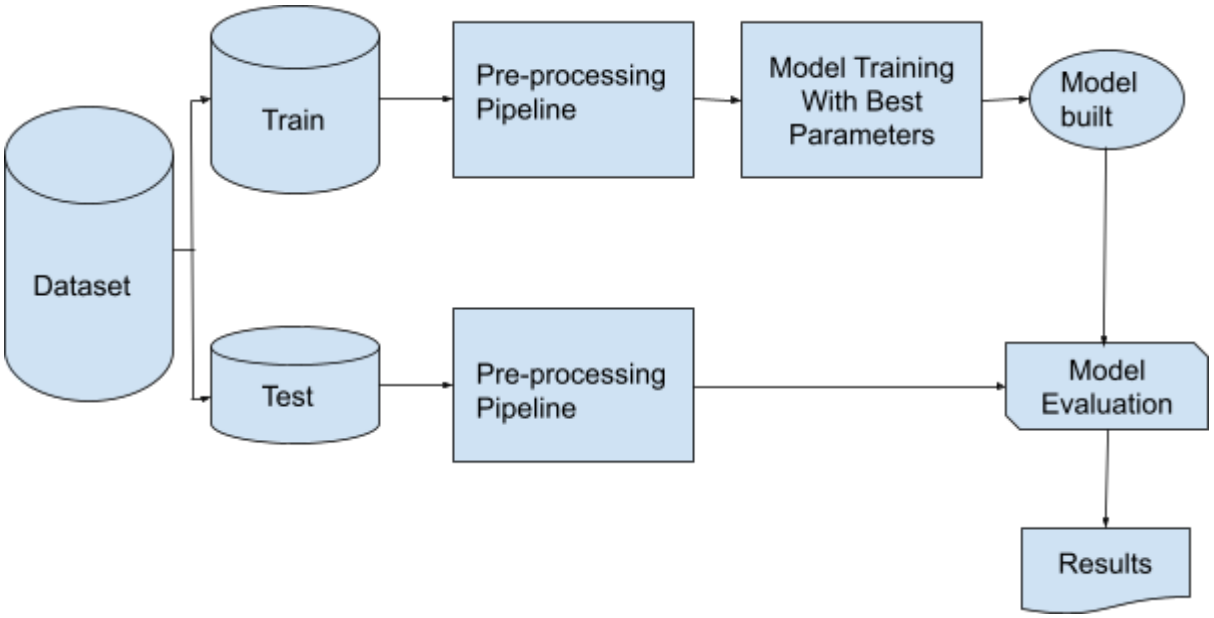
# 15 Tools and Technologies



- Visual Studio Code is used as an IDE.
- Machine Learning models are built using **scikit-learn**
- **API** is developed using Flask.
- UserInterface dashboard is developed using HTML and CSS
- Data is stored in the vector database ‘MongoDB’.
- GitHub is used as a version control system.
- ML-Flow is used to log experiments

# 16 Design Details

## 16.1 Process Flow




## 16.2 Event log

Should log each and every action and outcome throughout the project.

## 16.3 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.

# 17 Deployment

## 17.1 Deployment


# 18 Conclusion

Summarize the key design decisions, considerations, and expectations for the Credit Card Default Detection machine learning project. The HLD document serves as a reference for project development, deployment, and maintenance.