**Low Level Design (LLD)**

Credit Card Fraud Detection

**Revision Number: 1.8**

Agamdeep Singh

# Document Version Control

| Date Issued | Version | Description | Author |
| --- | --- | --- | --- |
| 21/05/2023 | 1.1 | First Draft | Agamdeep Singh |
| 23/05/2023 | 1.2 | Added Workflow chart | Agamdeep Singh |
| 26/05/2023 | 1.3 | Constraints, Additional Exception Scenarios | Agamdeep Singh |
| 29/05/2023 | 1.4 | Added KPIs | Agamdeep Singh |
| 1/06/2023 | 1.5 | Added user I/O flowchart | Agamdeep Singh |
| 8/06/2023 | 1.6 | Added EHR, LSTM model diagrams | Agamdeep Singh |
| 15/06/2023 | 1.7 | Added dataset overview and updated user I/O flowchart. | Agamdeep Singh |
| 17/06/2023 | 1.8 | Restructure and reformat LLD | Agamdeep Singh |

**Contents**

[Document Version Control 2](#_gjdgxs)

[**Abstract** 4](#_30j0zll)

[1 Introduction 4](#_tyjcwt)

[1.1 Why this Low-Level Design Document? 5](#_3dy6vkm)

[1.2 Scope 5](#_1t3h5sf)

[1.3 Constraints 5](#_2xcytpi)

[1.4 1.3 Risks 5](#_4d34og8)

[1.5 1.4 Out of Scope 5](#_2s8eyo1)

[2 Technical specifications 6](#_17dp8vu)

[2.1 Predicting Disease](#_26in1rg) 6

[2.2 Logging](#_lnxbz9) 7

[2.3 Database](#_35nkun2) 7

[3 Technology stack 8](#_1ksv4uv)

[4 Proposed Solution 8](#_44sinio)

[5 Model training/validation workflow 9](#_z337ya)

[6 User I/O workflow 9](#_3j2qqm3)

[7 Exceptional scenarios 9](#_1y810tw)

**Abstract**

In the twenty-first century, we live in the internet era. Nowadays, internet transactions have become a widespread way of payment, with credit cards being one of the most popular. However, there is a fear of being a victim of deception. Recently, there have been several fraudulent credit card transactions.

Within two days, we discovered 492 fraudulent transactions out of 284,807 transactions in European countries, according to the statistics.

It is critical that credit card companies detect fraudulent credit card transactions so that customers are not charged for products they did not purchase.

To address this grave issue, we attempted to develop a Web Application that uses machine learning techniques to detect these sorts of scams.

# Introduction

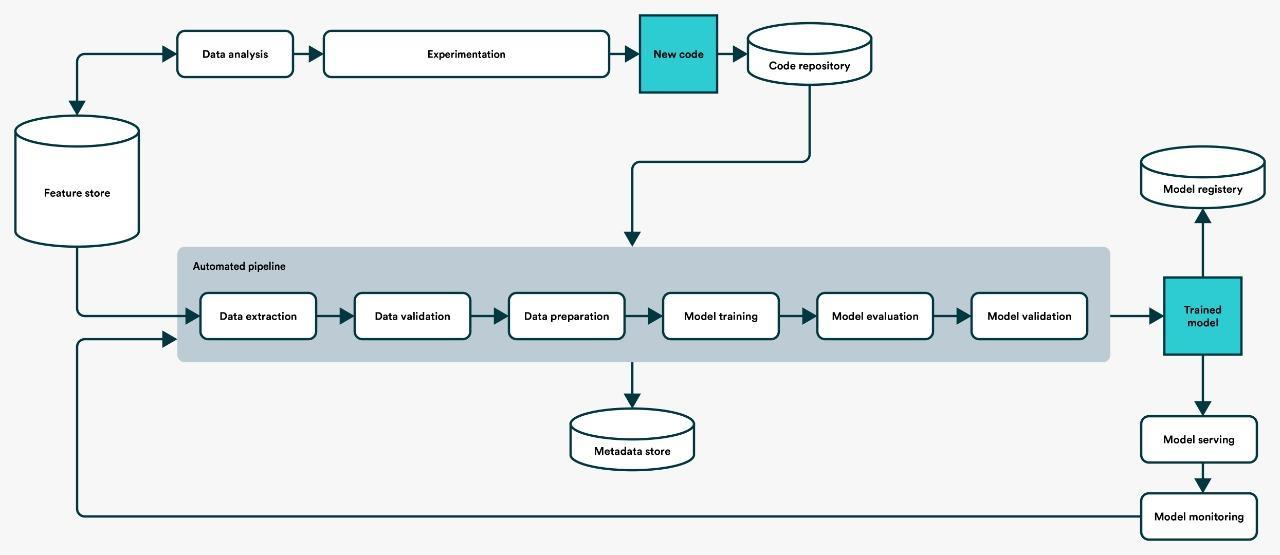
## Why this Low-Level Design Document?

This document's goal is to provide a clear overview of credit card fraud. It will explain the system's goal and characteristics, its interfaces, what the system will perform, the limitations under which it must function, and how the system will respond to external stimuli. This document is meant for both stakeholders and system developers, and it will be sent to upper management for approval.

**The main objective of the project is to predict whether a credit card transaction is fraud or not.**

Credit card fraud information, such as:

* **Time**
* **V1 to V27 features**
* **Amount**



## Scope

This software system will be implemented as a Web application. This technology will be designed to identify fraud and prevent clients from being paid for products they did not acquire.

## Risks

Specific dangers that have been discovered or should be evaluated should be documented.

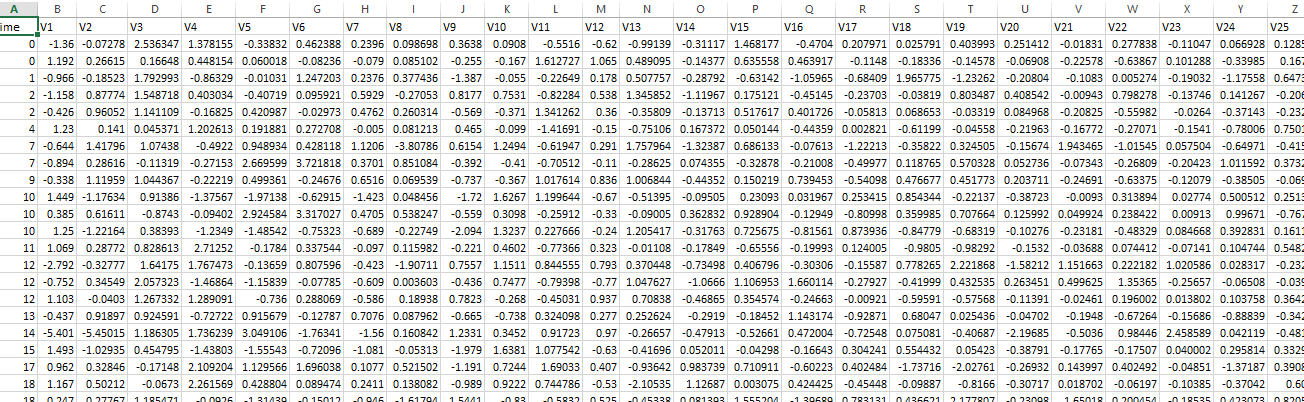
## Out of Scope

Define particular actions, skills, and objects that are not part of the project's scope.

## 2.1 Dataset

## 2.1.1 Credit Card dataset overview

Consists of one separate table, which contains credit card information and, most crucially, credit card historical data.The transcript table, on the other hand, contains credit card information.



## 2.1.2 Input schema

| **Feature name** | **Datatype** | **Null/Required** |
| --- | --- | --- |
| Time | float | Required |
| V1to V27 | float | Required |
| Time | float | Required |

## 2.2 Predicting Fraud

* The system displays whether fraud happened or not.
* The system presents the set of inputs required from the user.
* The user gives required information.
* The system should be able to predict whether fraud for the chosen transaction based on the user information.

## 2.3 Logging

We should be able to log every activity done by the user.

* The System identifies at what step logging required
* The System should be able to log each and every system flow.
* Developers can choose logging methods. You can choose database logging/ File logging as well.
* System should not be hung even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

## 2.4 Database

Every request must be stored in the database, and it must be stored in such a way that retraining the model is simple.

1. The User gives required information.

2. The system saves all data provided by the user or received on request to the database. You may pick your own database, such as MongoDB, MySQL, or Cassandra.

**2.5 Deployment**

Deployment is done using AWS



# Technology stack

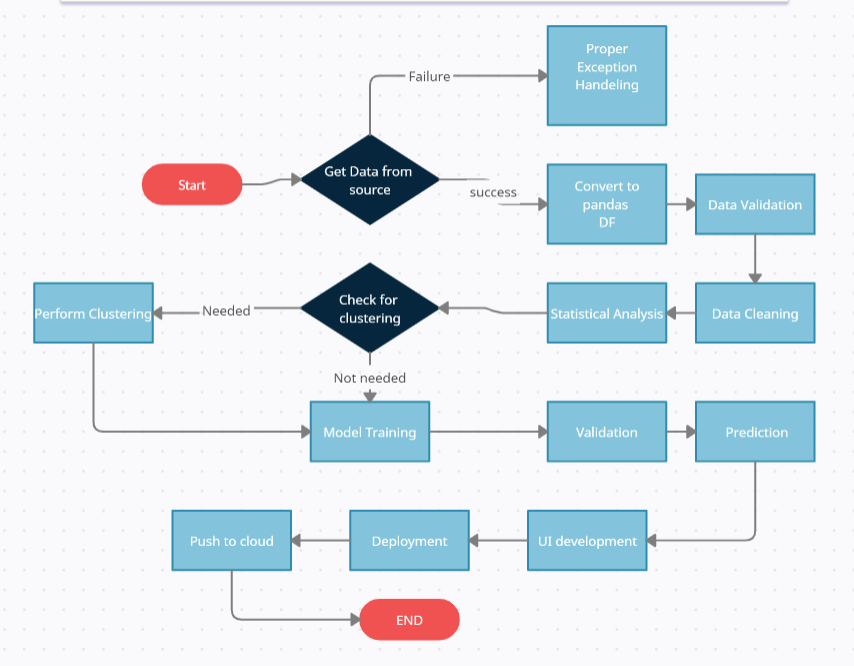
| **Front End** | **HTML/CSS/** |
| --- | --- |
| **Backend** | **Python Flask** |
| **Database** | **Cassandra** |
| **Deployment** | **AWS** |

**Proposed Solution-:**

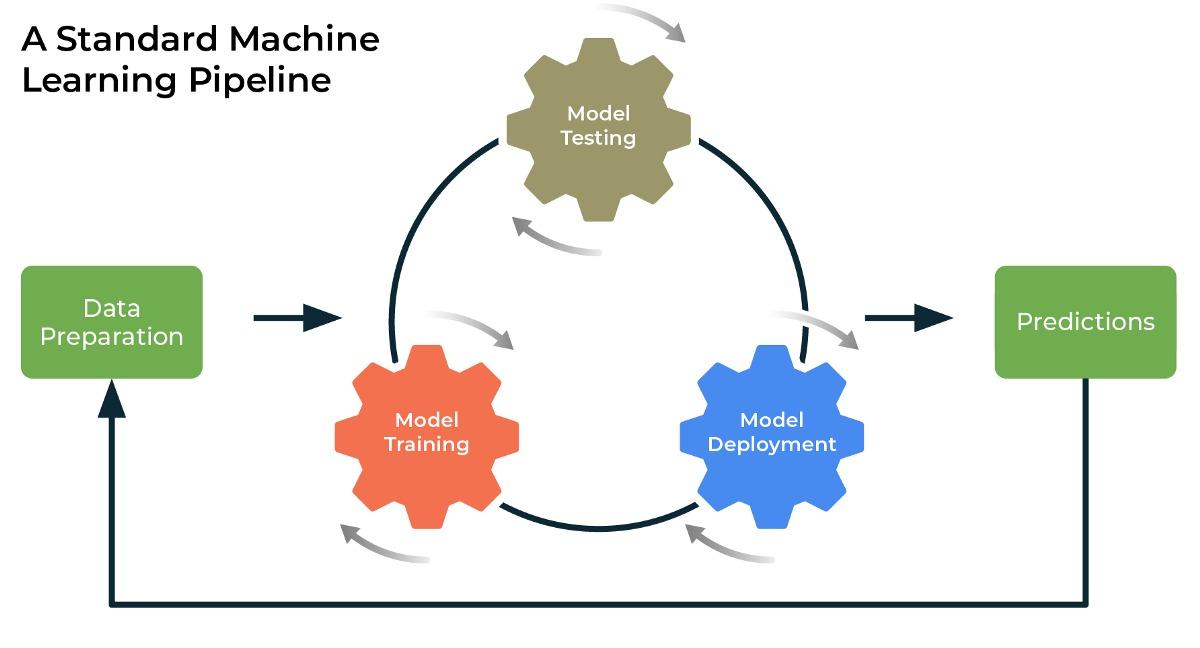
To prevent customers from fraud transactions.

1. **Baseline Model: Random Forest, since this is a classification problem.**
2. **Actual model: Random Forest.**

**Model training/validation workflow**



# User I/O workflow

****

# Exceptional scenarios

| Exception | Mitigation | Module |
| --- | --- | --- |
| 1.1 | First Draft | Agamdeep Singh |
| 1.8 | Last Draft- Added Workflow chart | Agamdeep Singh |

# Key performance indicators (KPI)

* Time and workload reduction using the Random Forest model.
* Comparison of accuracy of model prediction.