High Level Design (HLD)

Credit Card Default Prediction

Revision Number: 1.0
Last date of revision: 24/03/2024

Contents

Abstract

On demand of automation now a day business is involved in automating the complete life cycle of machine learning project. MLOPS is the frame which comes in to picture while talking about the automation of the Machine Learning project.

It is quite similar to DevOps but has slight difference in the deployment flow along with various more stages involving in it like the Retraining of Model after deployment provided if any drift with data, patterns are observed. MLOPs on cloud gives the leverage of carrying the automation of ML projects on various cloud platform like Azure, GCP and AWS.

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.

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 project description to represent a suitable model and coding for application. 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 aspects 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
- List and describe the non-functional attributes like:
  - Security
  - Reliability
  - Maintainability
  - Portability
  - Reusability
  - Application compatibility
  - Resource utilization
  - Serviceability

1.2 Scope

The HLD documentation presents the structure of the system, such as the 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.

1.3 Definitions

Term	*Description*
CC	Credit Card Defaulter Prediction
Database	Collection of all the information monitored by this system
IDE	Integrated Development Environment
AWS	Amazon Web Services

2 General Description

2.1 Product Perspective

CC system is a web application which will detect the defaulter and risk prediction of credit clients. It. No-SQL is used to retrieve, insert, delete, and update the database. Here the system stores each and every data in cassandra database.

2.2 Tools used

Python programming language and frameworks such as Numpy, Pandas, Scikit-learn are used to build the whole model.

- VS Code is used as IDE.
- For visualization of the plots, Matplotlib and Seaborn are used.
- AWS is used for deployment of the model.
- Astra DB(Cassandra) is used to retrieve, insert, delete, and update the database.
- Front end development is done using HTML/CSS
- Python Flask is used for backend development.
- GitHub is used as version control system.

2.3 Constraints

Constraints typically revolve around data availability, regulatory compliance, and model interpretability. Ensuring access to high-quality data while adhering to regulations such as GDPR and developing interpretable models within computational resource limitations are key considerations in this domain.

2.4 Assumptions

This may include expectations such as historical payment behaviour being indicative of future behaviour, the relevance of features such as credit utilization and payment history in predicting defaults, and the assumption that the data accurately reflects the underlying patterns of credit behaviour. These assumptions guide the model development process and are crucial for interpreting and applying the model's predictions effectively.

3 Design Details

3.1 Process Flow

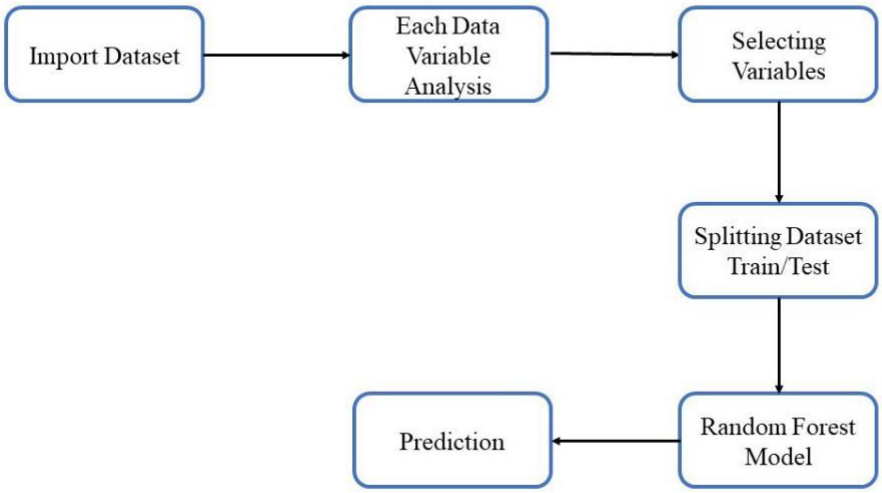

Figure 1: Process Flow of CC

3.2 Deployment Process

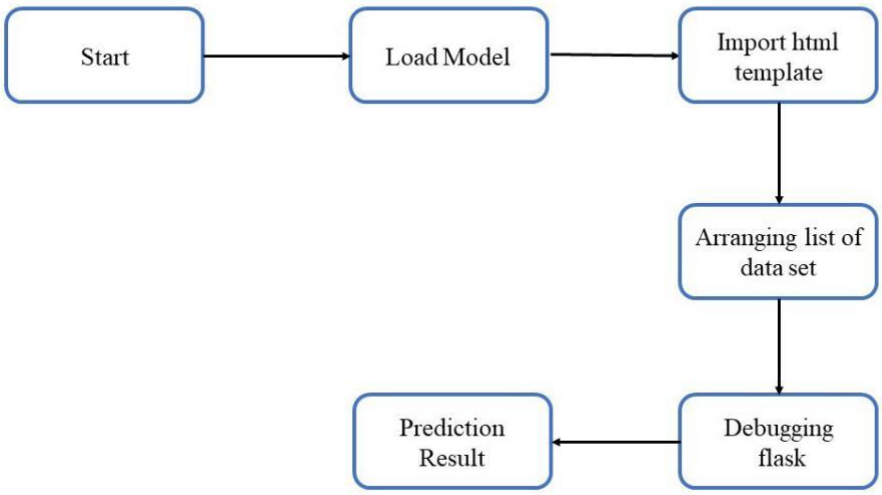

Figure 2: Web application architecture

The user interface is a very simple plain layout with little graphics. It will display information very clearly for the user and will primarily output information to the user through HTML pages. Also, all the details for the user input will be provided.

3.3 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 step logging 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 using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

3.4 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.

4 Performance

Performance will be measured at every instance so that if the model degrades over a period of time retraining approach is maintained.

4.1 Reusability

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

4.2 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.

4.3 Resource utilization

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

4.4 Deployment

5 References

1. https://drive.google.com/file/d/1BIaEtDPCWk6GQl_G9Idcw3pGWxxngrYn/view