Credit Card Default Prediction

Low Level Design (LLD)

B.Rajeswari
8th December 2023

Contents

Date Issued	Version	Description	Author
07 December 2023	1.0	LLD	B. Rajeswari

Abstract

Credit risk plays a major role in the banking industry business. Banks' main activities involve granting loan, credit card, investment, mortgage, and others. Credit card has been one of the most booming financial services by banks over the past years. However, with the growing number of credit card users, banks have been facing an escalating credit card default rate. As such data analytics can provide solutions to tackle the current phenomenon and management credit risks. This project discusses the implementation of an model which predicts if a given credit card holder has a probability of defaulting in the following month, using their demographic data and behavioral data from the past 6 months.

Introduction

1.1. Why this Low-Level Design Document?

The purpose of this document is to present a detailed description of the Deep EHR System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system and will be proposed to the higher management for its approval.

1.2. Scope

This software system will be a Web application. This system will be designed to predict the customers' probability in defaulting credit payments at earliest for better disease management and improved interventions using previous EHR records available. This system is designed to predict the credit card default from customers' information such as demographics, credit payment history etc.

2. Technical Specifications

2.1. Dataset

File Name	Finalized	Source
UCI_Credit_Card.csv	Yes	https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset

2.1.1. Dataset Overview

The data file consists of one table, UCI_Credit_Card, containing the personal information and historic data about the payments made in the previous 6 months (April to September, in this context), of about 30000 customers.

```
import pandas as pd
data = pd.read_csv('Credit_Card.csv')
```

```
data
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_A
0	1	20000.0	2	2	1	24	2	2	-1	-1	...	0.0	0.0	0.0	0.0	
1	2	120000.0	2	2	2	26	-1	2	0	0	...	3272.0	3455.0	3261.0	0.0	1
2	3	90000.0	2	2	2	34	0	0	0	0	...	14331.0	14948.0	15549.0	1518.0	1
3	4	50000.0	2	2	1	37	0	0	0	0	...	28314.0	28959.0	29547.0	2000.0	2
4	5	50000.0	1	2	1	57	-1	0	-1	0	...	20940.0	19146.0	19131.0	2000.0	36
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000.0	1	3	1	39	0	0	0	0	...	88004.0	31237.0	15980.0	8500.0	20
29996	29997	150000.0	1	3	2	43	-1	-1	-1	-1	...	8979.0	5190.0	0.0	1837.0	3
29997	29998	30000.0	1	2	2	37	4	3	2	-1	...	20878.0	20582.0	19357.0	0.0	
29998	29999	80000.0	1	3	1	41	1	-1	0	0	...	52774.0	11855.0	48944.0	85900.0	3
29999	30000	50000.0	1	2	1	46	0	0	0	0	...	36535.0	32428.0	15313.0	2078.0	1

30000 rows × 25 columns

Input Schema

Feature Name	Datatype	Null/Required
ID	Integer	Required
LIMIT_BAL	Integer	Required
SEX	Integer	Required
EDUCATION	Integer	Required
MARRIAGE	Integer	Required
AGE	Integer	Required

PAY_0	Integer	Required
PAY_2	Integer	Required
PAY_3	Integer	Required
PAY_4	Integer	Required
PAY_5	Integer	Required
PAY_6	Integer	Required
BILL_AMT1	Integer	Required
BILL_AMT2	Integer	Required
BILL_AMT3	Integer	Required
BILL_AMT4	Integer	Required
BILL_AMT5	Integer	Required
BILL_AMT6	Integer	Required
PAY_AMT1	Integer	Required
PAY_AMT2	Integer	Required
PAY_AMT3	Integer	Required
PAY_AMT4	Integer	Required
PAY_AMT5	Integer	Required
PAY_AMT6	Integer	Required
default.payment.next.month	Integer	Required

Predicting Credit Fault

- The system presents the set of inputs from the user.
- The user gives required information.

- The system should be able to predict whether the customer is likely to default in the following month.

2.3. Logging

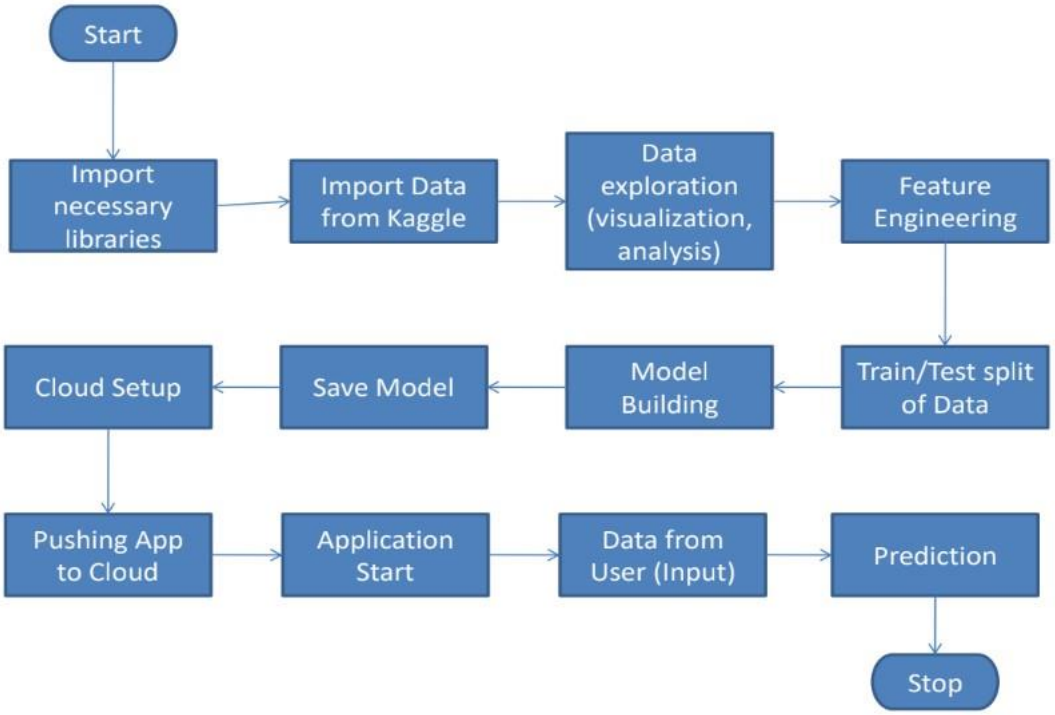
We should be able to log every activity done by the user.

- 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. Deployment

Deployed in Heroku.

3. Architecture



4. Architecture Description

4.1. Data Description

This dataset is taken from kaggle(url: https://www.kaggle.com/uciml/defaultof-

credit-card clients-dataset). It contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005.

Content

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)

4.2. Data Exploration

We divide the data into two types: numerical and categorical. We explore through each type one by one. Within each type, we explore, visualize and analyze each variable one by one and note down our observations. We also make some minor changes in the data like change column names for convenience in understanding.

4.3. Feature Engineering

Encoded categorical variables.

4.4. Train/Test Split

Split the data into 70% train set and 30% test set.

4.5. Model Building

Built models and trained and tested the data on the models.

Compared the performance of each model and selected the best one.

4.6. Save the model

Saved the model by converting into a pickle file.

4.7. Cloud Setup & Pushing the App to the Cloud

Selected Heroku for deployment. Loaded the application files from Github to Heroku.

4.8. Application Start and Input Data by the User

Start the application and enter the inputs.

4.9. Prediction

After the inputs are submitted the application runs the model and makes predictions. The out is displayed as a message indicating whether the customer whose demographic and behavioral data are entered as inputs, is likely to default in the following month or not.

Test Case Description	Pre-Requisite	Expected Result
Verify whether the Application URL is accessible to the user	1. Application URL should be defined	Application URL should be accessible to the user
Verify whether the Application loads completely for the user when the URL is accessed	1. Application URL is accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify whether user is able to see input fields on logging in	1. Application URL is accessible 2. Application is deployed	User should be able to see input fields on logging in
Verify whether user is able to edit all input fields	1. Application URL is accessible 2. Application is deployed	User should be able to edit all input fields

Verify whether user gets Submit button to submit the inputs	1. Application URL is accessible 2. Application is deployed	User should get Submit button to submit the inputs
Verify whether user is presented with recommended results on clicking submit	1. Application URL is accessible 2. Application is deployed	User should be presented with recommended results on clicking submit