

# CREDIT CARD DEFAULT PREDICTION

---

Architecture

**Manisha Singh, August 2023**

## Contents

1. Introduction	3
1.1. What is Low-Level design document?	3
1.2. Scope	3
2. Architecture	4
3. Architecture Description	5
3.1. Data Description	5
3.2. Data Exploration	6
3.3. Feature Engineering	6
3.4. Train/Test Split	6
3.5. Model Building	6
3.6. Save the Model	6
3.7. Cloud Setup & Pushing the App to the Cloud	6
3.8. Application Start and Input Data by the User	6
3.9. Prediction	6
4. Unit Test Cases	7

## 1. Introduction

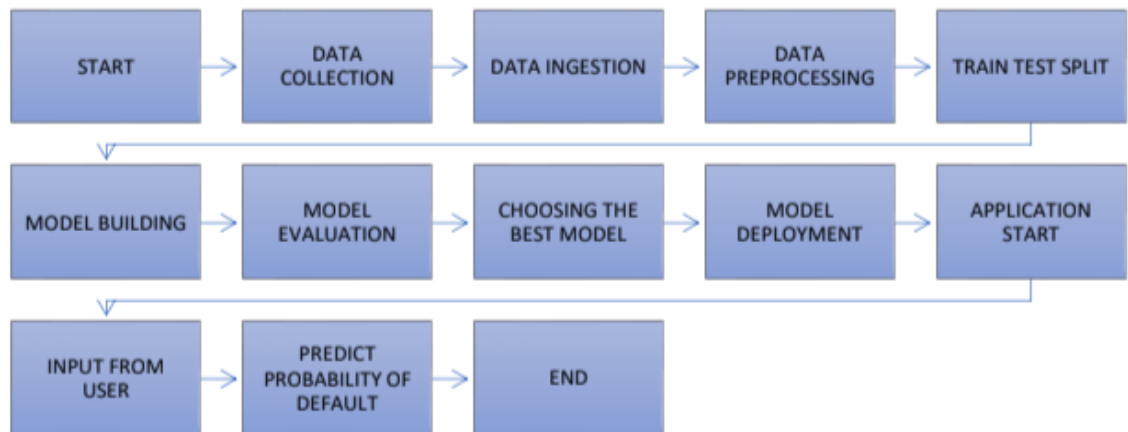
### 1.1. Why the Low-Level Design Document?

The goal of LLD or a Low-Level Design document is to give the internal logical design of the actual program code. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

### 1.2. Scope

Low-level design (LLD) is a component-level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement analysis and then refined during data design work.

## 2. Architecture



## 3. Performance

### 3.1. Data Description

This dataset 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.

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)

### 3.2. 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.

### **3.3. Feature Engineering**

Encoded categorical variables.

### **3.4. Train/Test Split**

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

### **3.5. Model Building**

Built models, trained, and tested the data on the models. Compared the performance of each model and selected the best one.

### **3.6. Save the Model**

Saved the model by converting into a pickle file.

### **3.7. Application Start and Input Data by the User**

Start the application and enter the inputs.

### **3.8. Prediction**

After the inputs are submitted the application runs the model and makes predictions. The output 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.

## 4. Unit Test Cases

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.