Low Level Design (LLD)

**Credit Card Default Prediction**

***PARAS DAHIYA***

**ABSTRACT**

In the dynamic landscape of financial risk assessment, the challenge of predicting credit defaults for commercial banks shares conceptual parallels with classification tasks in diverse domains. This project employs classical machine learning techniques encompassing Data Exploration, Cleaning, Feature Engineering, Model Building, and Testing to craft a robust solution. The goal is to predict the probability of credit default based on the distinctive characteristics and payment histories of credit card owners.

.

**CONTENTS**

Title Page i

Abstract ii

Content iii

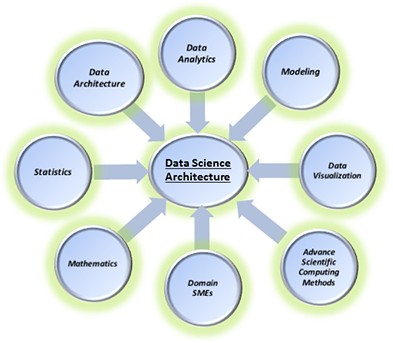
|  |  |  |
| --- | --- | --- |
|  | Introduction | 1 |
|  | Why this LLD Documentation? | 1 |
| 1 | Architecture | 1 |
| 2 | Architecture Design | 1 |
| 2.1 | Data Gathering from Main Source | 2 |
| 2.2 | Data Description | 2 |
| 2.3 | Import Data into Database | 2 |
| 2.4 | Export Data from Database | 2 |
| 2.5 | Data Pre-Processing | 3 |
| 2.6 | Modelling | 3 |
| 2.7 | UI Integration | 3 |
| 2.8 | Data From User | 3 |
| 2.9 | Data Validation | 4 |
| 2.10 | Rendering the Result | 4 |
| 3 | Deployment | 4 |
| 3.1 | Unit Test Case | 4 |
|  |  |  |

**Introduction**

# Why this Low-Level Design Document?

The main purpose of this LLD documentation is to feature the required details of the project and supply the outline of the machine learning model and also the written code. This additionally provides the careful description on However the complete project has been designed end-to-end.

# Architecture



**2 Architecture Design**

For this project, we have implemented the machine learning life cycle to create a basic web application which will predict the default case by applying machine learning algorithms to available data using python libraries like Pandas, NumPy, Matplotlib, seaborn and sklearn.

# 2.1 Data Gathering

The data for the current project is being gathered from Kaggle dataset, the link to the data is:

[Default of Credit Card Clients Dataset | Kaggle](https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset)

## 2.2 Data Description

Data Description: This dataset describes credit card data. They are classified into: Default payment or not.

1. ID: ID of each client
2. LIMIT\_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit
3. SEX: Gender (1=male, 2=female)
4. EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
5. MARRIAGE: Marital status (1=married, 2=single, 3=others)
6. AGE: Age in years
7. 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)
8. PAY\_2: Repayment status in August, 2005 (scale same as above)
9. PAY\_3: Repayment status in July, 2005 (scale same as above)
10. PAY\_4: Repayment status in June, 2005 (scale same as above)
11. PAY\_5: Repayment status in May, 2005 (scale same as above)
12. PAY\_6: Repayment status in April, 2005 (scale same as above)
13. BILL\_AMT1: Amount of bill statement in September, 2005 (NT dollar)
14. BILL\_AMT2: Amount of bill statement in August, 2005 (NT dollar)
15. BILL\_AMT3: Amount of bill statement in July, 2005 (NT dollar)
16. BILL\_AMT4: Amount of bill statement in June, 2005 (NT dollar)
17. BILL\_AMT5: Amount of bill statement in May, 2005 (NT dollar)
18. BILL\_AMT6: Amount of bill statement in April, 2005 (NT dollar)
19. PAY\_AMT1: Amount of previous payment in September, 2005 (NT dollar)
20. PAY\_AMT2: Amount of previous payment in August, 2005 (NT dollar)
21. PAY\_AMT3: Amount of previous payment in July, 2005 (NT dollar)
22. PAY\_AMT4: Amount of previous payment in June, 2005 (NT dollar)
23. PAY\_AMT5: Amount of previous payment in May, 2005 (NT dollar)
24. PAY\_AMT6: Amount of previous payment in April, 2005 (NT dollar)
25. default.payment.next.month: Default payment (1=yes, 0=no)

## 2.3 Import Data into Database

Created an api for the upload of the data into the SQL database, steps performed are:

* + - Connection is made with the database.
    - Created a database with name Prediction.
    - SQL command is written for creating the data table with required parameters.
    - And finally, a SQL command is written for uploading the dataset into the data table by bulk insertion.

## 2.4 Export Data from Database

In the above created api, the download url is also being created, which downloads the data into a csv file format.

# 2.5 Data Pre-Processing

Steps performed in pre-processing are:

* First the data types are being checked
* Replaced such values with numpy "nan" so that we can handle the missing values.
* Performed categorical imputing for the required columns.
* Added quote on the the whole data.

And, the data is ready for passing to the machine learning algorithm.

# 2.6 Modelling

After selecting the features which are more correlated to price the next step involves applying machine algorithms and creating a model. As our dataset consist of labelled data, we will be using supervised machine learning algorithms. The machine learning algorithms that we will be using in our project are:

Naïve Bayes

XGBOOST

# 2.7 UI Integration

The front of the application will be created using the HTML, CSS , Bootstrap framework where users will have the functionality of entering their flight data. This data will be sent to the back-end service where the model will predict the output according to the provided data. The predicted value is sent to the front-end and displayed.

# 2.8 Data from User

The data from the user is retrieved from the created HTML web page.

# 2.9 Data Validation

The data provided by the user is then being processed by app.py file and validated. The validated data is then sent for the prediction.

# 2.10 Rendering Result

The data sent for the prediction is then rendered to the web page.

# 3. Deployment

After getting the model with the best accuracy we store that model in a file using the pickle module. The back-end of the application will be created using Flask Framework where API end-points such as GET and POST will be created to perform operations related to fetching and displaying data on the front-end of the application.

# 3.1 Unit Test

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Expected Result** | **Actual Result** | **Status** |
| User Interface | Verify whether a user is able to give input to all fields. | As Expected | Pass |
| UserInterface URL | verify whether hosted URL is properly accessible or not | As Expected | Pass |
| UserInterface URL | Verify whether the user Interface loads completely for the user when the URL is accessed | As Expected | Pass |