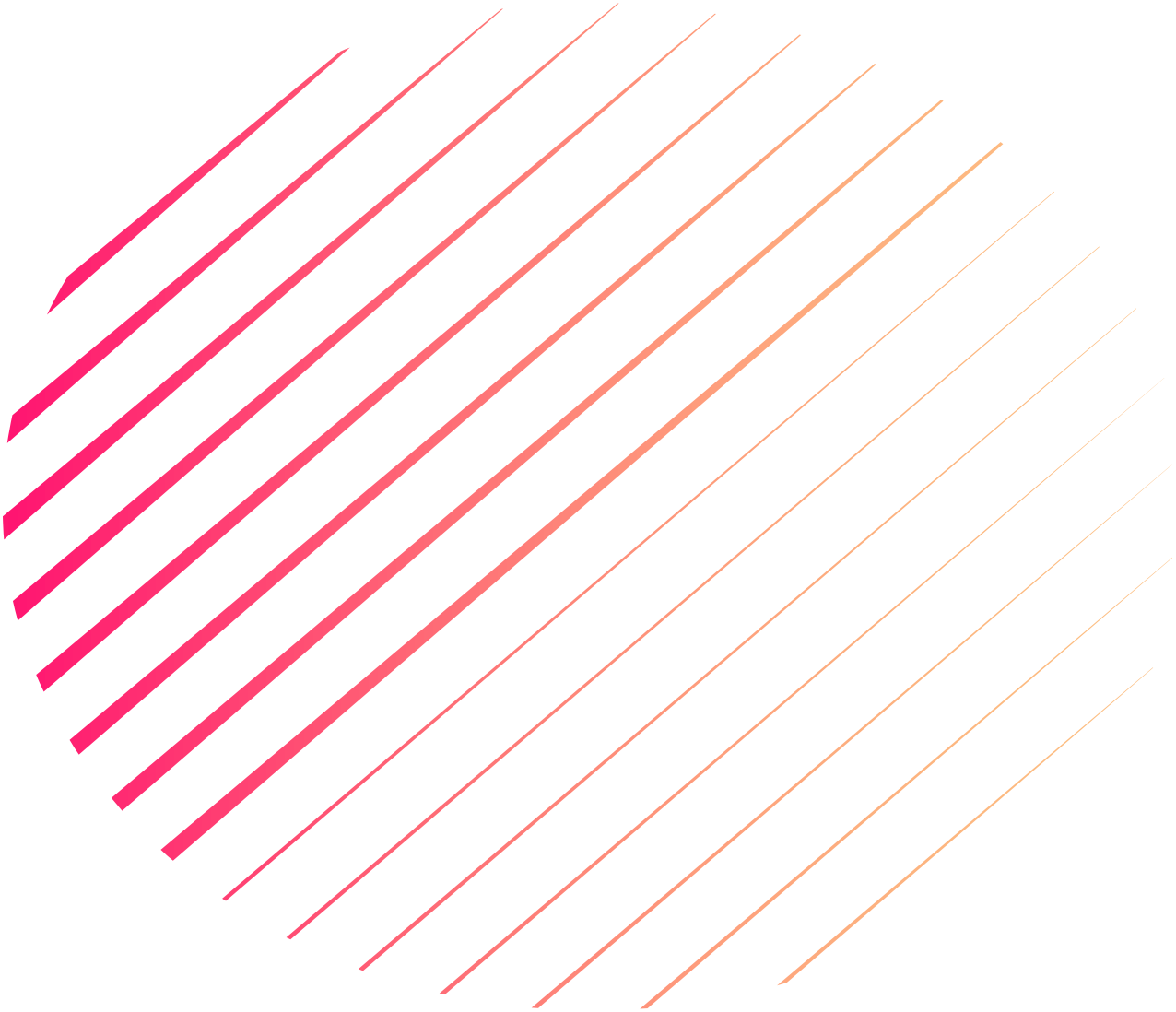
Credit Card Default Project

Low Level Document (LLD)

HARSHA LOURDU M



Contents

Abstract 2
1 Introduction 3
1.1 What is Low-Level Design Document? 3
1.2 Scope 3
1.3 Definitions 3
2 Architecture 4
3 Architecture description 4
3.1 Data Collection 4
3.2 Data Ingestion 5
3.3 Data Pre-Processing 5
3.4 Model Building 6
3.5 Metrics Used for Model Evaluation 6
3.6 Saving the Best Model 6
3.7 User Interface using Dash 7

Abstract

Financial threats are displaying a trend about the credit risk of commercial banks as the incredible improvement in the financial industry has arisen. In this way, one of the biggest threats faced 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.

Use of various classification models like Decision Tree, Random Forest, XGBoost, MLPClassifier was done. XGBoost was selected as best model from above. The XGBoost model provided 82.63% accuracy in training. Testing accuracy for model was 82.3%. 80% data was used for training and 20% data was used for testing.

After Providing various details the model will predict whether customer will default next month ornot.

1 Introduction

1.1 What is Low-Level Design Document?

The goal of LLD or a Low-level design document is to give an internal logical design of the actual program code for the Credit Card Default Probability Prediction. LLD describes the class diagrams with the methods and relations between classes and the program specs. It describes the modules so that the programmer can directly code the program from the document.

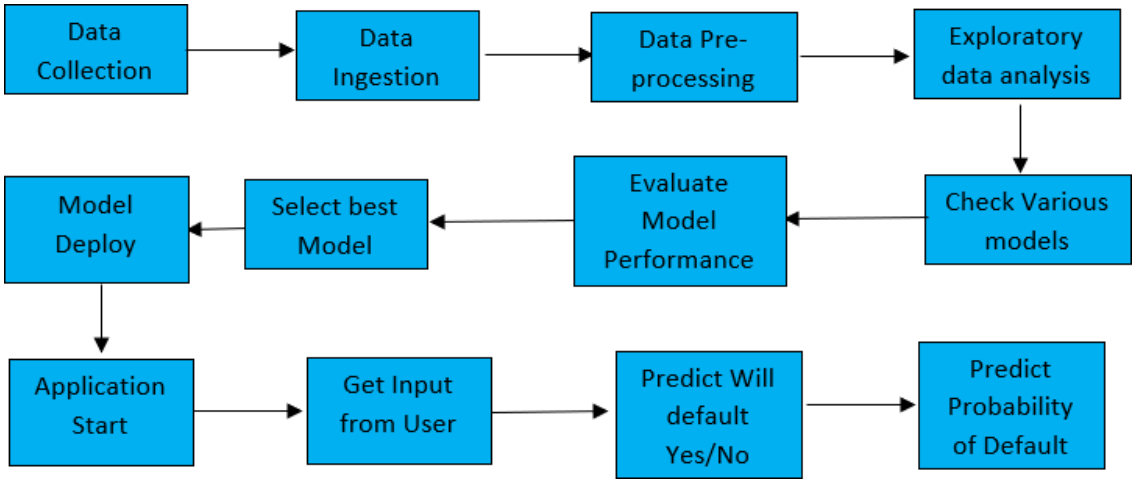
1.2 Scope

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 defined during data design work.

1.3 Definitions

Term	Description
IDE	Integrated Development Environment
EDA	Exploratory Data Analysis
XGBoost	Extreme Gradient Boost Algorithm
ML	Machine Learning
MLP Classifier	Multi-Layer Perceptron Classifier
KNN	K-Nearest Neighbours
VS Code	Visual Studio Code

2 Architecture



3 Architecture description

3.1 Data Collection

For training and testing the model I used a publicly available dataset on Kaggle. 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.

URL - https://www.kaggle.com/datasets/uciml/default-of-credit-card-clients-dataset

Data Information

- 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)

Variable Information

This is a classification problem in which default.payment.next.month is the target variable. The Aim of this project is to predict whether customer will default or not and predict the probability of default. Various attributes are given below:

Variable Name	Measurement Unit	Description
LIMIT_BAL	NT Dollar	Amount of given credit
SEX	Integer	Gender (1=male, 2=female)
EDUCATION	Integer	1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown
MARRIAGE	Integer	Marital status (1=married, 2=single, 3=others)
AGE	Years	Age of the person in years
PAY_0-6	Integer	Repayment status for various months
BILL_AMT1-6	NT Dollar	Amount of billed statements for various months
PAY_AMT1-6	NT Dollar	Amount of Previous payments done
default.payment.next.month	Binary	Will Customer default? Yes/No

3.2 Data Ingestion

For loading dataset in coding environment, I used Pandas library in python. Dataset is loaded using pandas.read_csv function of the library.

3.3 Data Pre-Processing

Once the Data is Ingested using Pandas. I used below steps for data pre-processing

- Drop the column which is statistically not Important, here I dropped the ID column
- Convert variables: SEX, EDUCATION, MARRIAGE into object as they are categortical variables
- Separate Categorical and Continuous variable.
- For Continuous variables Scale the model with StandardScaler or MinMaxScaler if necessary for model. Scaling is not necessary for tree-based models.
- Perform One-Hot Encoding on Categorical variables
- Join Continuous and One Hot Encoded Variables
- Data Pre-processing is done.

3.4 Model Building

I have built Decision Tree Classifier, Random Forest Classifier, XGBoost Classifier, KNN Classifier and MLP Classifier.

Each of above models were built in separate notebooks and were evaluated.

To evaluate the model, I split the data into 80% training and 20% testing using train_test_split function. Then I evaluated various metrics on training and testing data.

4-fold Cross validation was also done on training Data for Hyper Parameter tuning and model generalisation.

3.5 Metrics Used for Model Evaluation

I used Accuracy score and Classification report for choosing the best model. I

also used Classification report for evaluating the model.

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives, and False Negatives are used to predict the metrics of a classification report.

3.6 Saving the Best Model

The Best Model was XGBoost Classifier and it also took lesser time in training than Random Forest Algorithm.

Training Accuracy	0.8236
Testing Accuracy	0.8230

Below is Classification report for test dataset

	precision	recall	f1-score	support
0	0.84	0.96	0.89	4654
1	0.71	0.36	0.48	1346
accuracy			0.82	6000
macro avg	0.77	0.66	0.69	6000
weighted avg	0.81	0.82	0.80	6000

Parameters	Value
n_estimators	500
Learning_rate	0.01
max_depth	4

3.7 User Interface using Dash