iNeuron

ARCHITECTURE

CREDIT CARD DEFAULT

PROBABILITY PREDICTION

October, 2022

Written by Namdeo Patil

CONTENTS

DOCUMENT VERSION CONTROL

Date Issued	Versions	Description	Author
18/10/2022	1.0.1	First Draft	Namdeo Patil
20/10/2022	1.0.2	Added Technical specifications	Namdeo Patil
20/10/2022	1.0.3	Added Technology stack, Proposed solution and Workflow	Namdeo Patil

ABSTRACT

Financial threats are displaying a trend in the credit risk of commercial banks as the incredible improvement in the financial industry has arisen. 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 the credit card owner's characteristics and payment history.

With the help of Data Science and Machine learning technology, I developed an application, which allows a banker to determine the probability Of Default in just a few second

1) Introduction

1.1 Why this 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 Concrete Compressive Strength Prediction System. 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

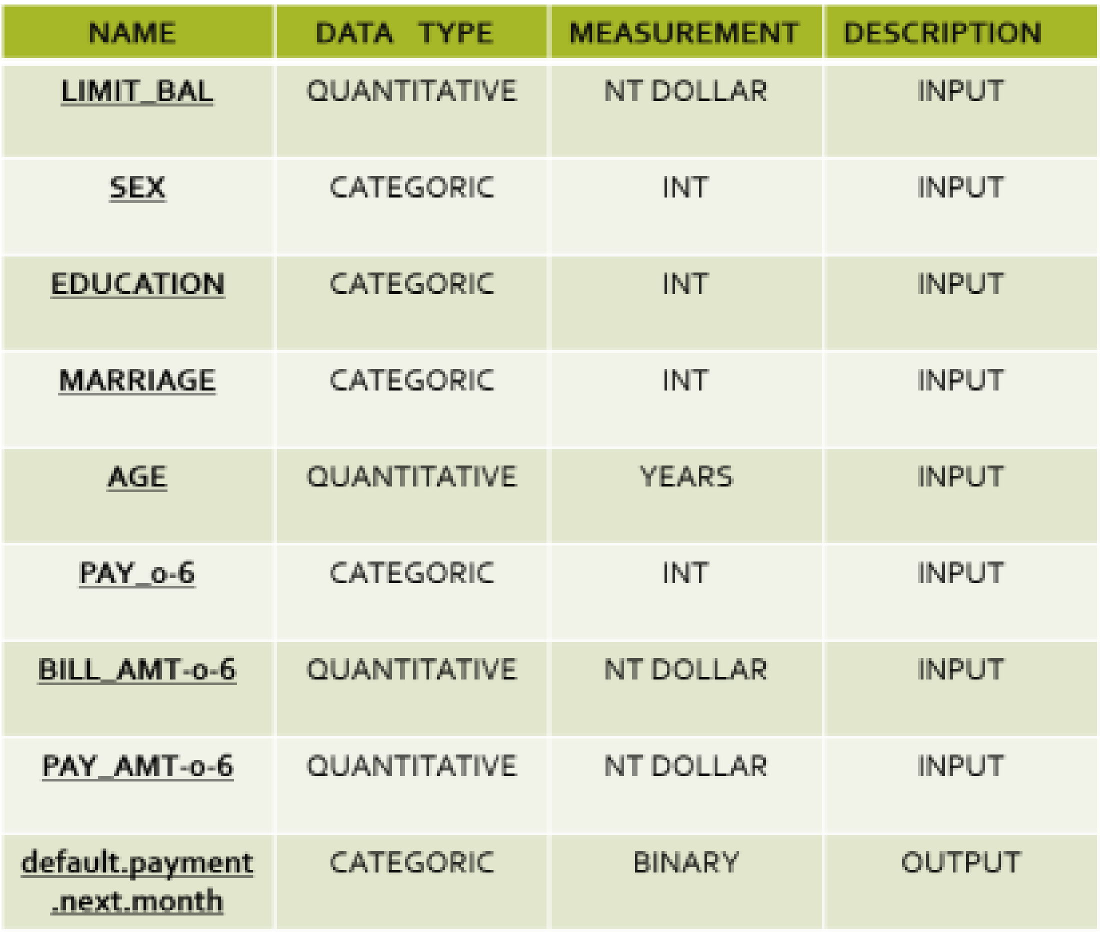
Low-level design (LLD) is a component level design process that follows a step-by-step refinement process. This process can be used for designing data structures, required software architecture, source code and ultimately, performance algorithms. Overall, the data organization may be defined during requirement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

2) Technical Specifications

2.1 Dataset Overview

For training and testing the model, I used the public data set available in Kaggle, “Default of Credit Card Clients Dataset” by UCI.

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

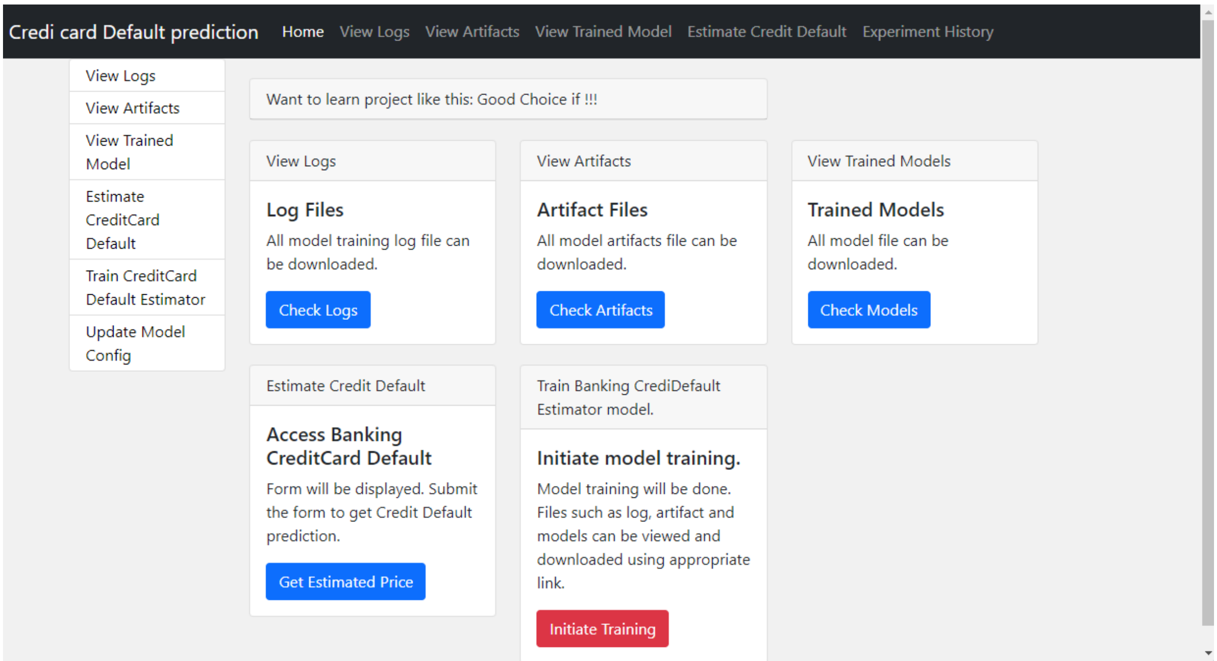


NAME	DATA TYPE	MEASUREMENT	DESCRIPTION
LIMIT_BAL	QUANTITATIVE	NT DOLLAR	INPUT
SEX	CATEGORIC	INT	INPUT
EDUCATION	CATEGORIC	INT	INPUT
MARRIAGE	CATEGORIC	INT	INPUT
AGE	QUANTITATIVE	YEARS	INPUT
PAY_0-6	CATEGORIC	INT	INPUT
BILL_AMT-0-6	QUANTITATIVE	NT DOLLAR	INPUT
PAY_AMT-0-6	QUANTITATIVE	NT DOLLAR	INPUT
default.payment .next.month	CATEGORIC	BINARY	OUTPUT

2.2 Predicting Credit card default probability

The web application must be loaded properly for the users without any technical glitches like server timeouts.

- **It must display the input fields and the "Predict" button to the users who accessed the application and allow the user to enter the values with respect to the attributes of the customer.**
  - **The user gives the required information.**
- **Then the application should be able to predict the probability of default based on the information given by the user about the customer.**



Credi card Default prediction Home View Logs View Artifacts View Trained Model Estimate Credit Default Experiment History

View Logs
View Artifacts
View Trained Model
Estimate CreditCard Default
Train CreditCard Default Estimator
Update Model Config

Want to learn project like this: Good Choice if !!!

View Logs
Log Files
All model training log file can be downloaded.
Check Logs

View Artifacts
Artifact Files
All model artifacts file can be downloaded.
Check Artifacts

View Trained Models
Trained Models
All model file can be downloaded.
Check Models

Estimate Credit Default
Access Banking CreditCard Default
Form will be displayed. Submit the form to get Credit Default prediction.
Get Estimated Price

Train Banking CrediDefault Estimator model.
Initiate model training.
Model training will be done. Files such as log, artifact and models can be viewed and downloaded using appropriate link.
Initiate Training

2.3 Logging

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

- **The system should be able to log every step in the program flow.**
- **System should not be hung even after using so many loggings.**
  - **Logging makes debugging much easier, like we can directly go to that specific line of code, having bugs. • In this project, logs will be written in the files**
- **"development_logs.log" and the "deployment_logs.log" respectively.**

2) Technology stack

Front-end	HTML with CSS styling
Back-end	Python version 3.7, Flask version 2.0.1
Deployment	Heroku, gunicorn version 20.1.0

3) Proposed solution

The solution proposed here is a web application, which takes the details of the customer and those details will be taken by a machine learning model in the backend, which will then predict the probability of default and display it on the front-end page of the user.

4) Workflow

START DATA -→ INGESTION DATA

-→ Data validation-→MODEL Transformation

-→ Model Trainer -→Model EVALUATION

-→BEST MODEL-→ MODEL Pusher

-→DEPLOYMENT-→ APPLICATION START

-→INPUT FROM USER

-→PREDICT PROBABILITY OF DEFAULT -→ END