LOW LEVEL DESIGN(LLD)

CREDIT CARD DEFAULT PREDICTION

Ineuron Internship

Dhruv Bakshi

23/03/2023

Contents

Abstract

In these uncertain financial times managing risk is becoming utmost priority of financial institutions. Banks and credit card companies face the impact of users defaulting on their payments now and then. Some users may default on payment once or twice due to some genuine reasons (like health emergency, loss of job etc.) and repay later. Some others may default payment continuously. Hence there is need to predict whether the user will default on payments based on his/her previous financial history and demographic data. Machine learning model tries to provide a solution to this problem.

1.Introduction

1.1. Why this Low-Level design document?

The goal of LLD or a Low-Level Design (LLD) document is to give the internal logical design of the actual program code for Credit card default prediction. 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.

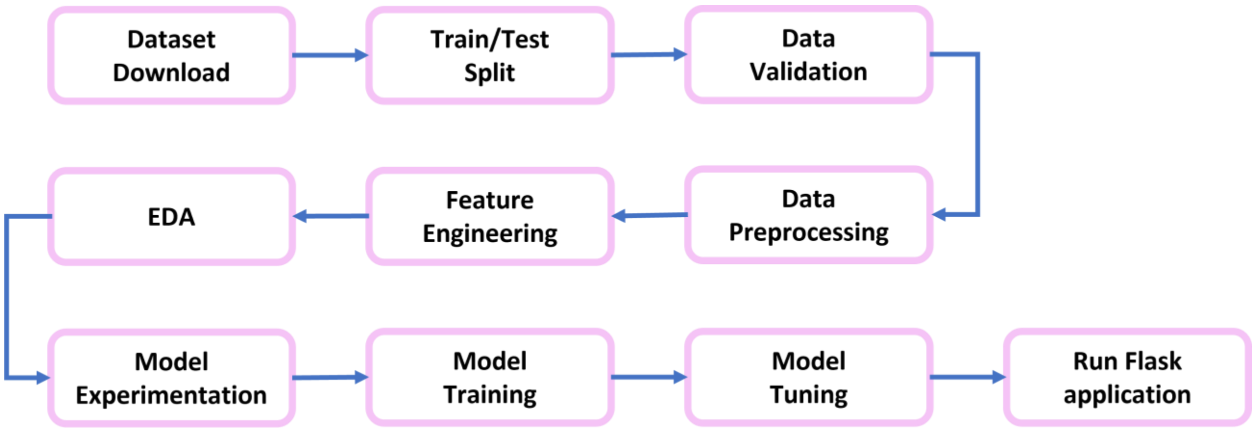
1.2. Scope

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.Problem Statement

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 faces 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.

3.Architecture Description



3.1. Dataset Download : Data was downloaded from Kaggle (https://www.kaggle.com/uciml/defaultof-credit-card-clients-dataset) using Kaggle API. The 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. The dataset had 24 independent variable and 1 target variable.

3.2. Train/Test Split : Data was split into train and test using StratifiedShuffleSplit function imported from Sklearn library. New category column consisting of four bins (ranging from 0 to max) was created based on existing column LIMIT_BAL . Data was split in ratio **80:20** based on this new category column, 80% constituted train dataset and 20% test dataset. The new category column was deleted from both test and train datasets. Train dataset was used to train machine learning models and Test dataset was used for comparison with predicted results.

3.3. Data Validation : A schema file with extension. yaml was created. Both test and train datasets were compared with schema and validated. Using Evidently library datadrift report is generated in html and json format.

3.4. Data Preprocessing : The process included checking of null values, cleaning of column (EDUCATION, MARRIAGE, PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6) values and renaming target column in both train and test datasets.

3.5. Feature Engineering : The process includes deleting columns (ID, AGE) from both datasets. Then performing smote resampling using Imblearn.oversampling library to balance the datasets. Drop duplicates. Saving the datasets in csv format. Performing onehot encoding on categorical columns and standard scalar on numerical columns with the help of column transformer. Saving both datasets in numpy array format.

3.6. EDA : Using pandas profiling library generate test and train eda reports in html format.

3.7. Model Experimentation : Dagshub account is used to get mlfow remote uri. Using Mlflow library various experimentations are performed with different combination of ML models. Different log parameters and log metrics of models are stored. Model files are not stored.

3.8. Model Training : Three Sklearn models namely LogisticRegression, KNeighborsClassifier and RandomForestClassifier are created and their scores and comparison with base accuracy is stored in csv.

3.9. Model Tuning : The best performing model i.e. RandomForestClassifier is further tuned using GridSearchCV. The best parameters and scores are saved. The model has accuracy of **83%**. Model is build using the best parameter. Model is stored as pickle file.

3.10. Run Flask application : Model is deployed in the flask application. The flaks application takes inputs from users and predict the result using this model. The flask application also maintains previous records in csv format.