LOW LEVEL DESIGN

Credit Card Default Prediction

|  |  |
| --- | --- |
| Written By | B.Sreenath Reddy |
| Document Version |  |
| Last Revised Date |  |

Document Control

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Written By | Comments |
| 0.1 |  | B.Sreenath Reddy | Introduction & Architecture defined |
| 0.2 |  | B.Sreenath Reddy | Architecture & Architecture Description appended and updated |
| 0.3 |  | B.Sreenath Reddy | Unit Test Cases defined and appended |

Contents

1 Introduction-------------------------------------------------------------------------4 1.1 What is Low-Level Design Document--------------------------------------4 1.2 Scope------------------------------------------------------------------------------4

2 Architecture--------------------------------------------------------------------------5

3 Architecture Description--------------------------------------------------------- 6 3.1 Data Description ---------------------------------------------------------------6 3.2 Export Data from DB to CSV for training----------------------------------6 3.3 Data Pre-processing -------------------------------------------------------------6 3.4 .Feature Engineering --------------------------------------------------------6

3.5 Train/Test Split----------------------------------------------6

3.6 Hyperparameter Tuning-------------------------------------------------------7

3.7 Selecting Model ---------------------------------------------------------------------7 3.8 Save Model -----------------------------------------------------------------------7 3.9 Deployment --------------------------------------------------------------7

1.Introduction

1.1 What is Low-Level design document?

The goal of LLD or a low-level design document (LLD) is to give the internal logical design of the actual program code for Thyroid Disease Detection System. LLD describe the class diagrams with the methods and relations between classes and program specs. It describe 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 :

**Data Pre-processing**

**Export data from database to csv for training**

**Start**

**Model saving**

**Hyperparameter Tuning**

**Get best model for each classification**

**Data Classification**

**Data from client to be predicted**

**Application Start**

**Pushing app to cloud**

**Cloud Setup**

**Data Pre-processing**

**Data Classification**

**Model call for specific cluster**

**Prediction**

**Export data from database to csv to prediction**

**End**

**Export prediction to csv**

3. Architecture Description

3.1 Data Description :

The dataset was taken from Kaggle (URL: https://www.kaggle.com/uciml/defaultof-credit-card-clients-dataset), 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

3.2 Export Data from database to CSV for Training

Here we will be exporting all batches of data from database into one csv file for training.

3.3 Data Pre-processing :

This included importing of important libraries such as seaborn, matplotlib, pandas etc. We imported the same dataset mentioned above from Kaggle. We first explore our data set in Jupyter Notebook and decide what pre-processing and Validation we have to do such as imputation of null values, dropping some column, etc and then we have to write separate modules according to our analysis, so that we can implement that for training as well as prediction data.

3.4. Feature Engineering:

Merging 2 or mode columns to get indepth knowledge and information regarding the data.

3.5. Train/Test Split:

This library was imported from Sklearn to divide the final dataset into the ratio of 80-20%, where 80% of the data was used to train the model and the latter 20% was used to predict the same.

3.6 Hyperparameter Tuning :

After selecting best model for each cluster, we will do hyperparameter tuning for each selected model, and try to increase performance of the models

3.7 Selecting Model :

We tried and tested multiple models such as LogisticRegression, RandomForest ,Decision Tree, for the model and came up with the model with the best performance, i.e the Random Forest Classifier.

3.8 Save Model :

Model was saved using the pickle library which saves the file in a binary mode.

3.9. Deployment :

We will be deploying the model to AWS.