LOW LEVEL DESIGN (LLD)

Predict Credit Risk Using South German Bank Data

Revision Number : 1.0

Last Date of Revision : 05/10/2021

Document Control

Version	Date	Author	Comments
1.0	21/08/2021	Shrey	Document Created

CONTENT

1. Introduction

1.1 What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Food Recommendation 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 refined during data design work.

2. Architecture

2.1 Model Flow

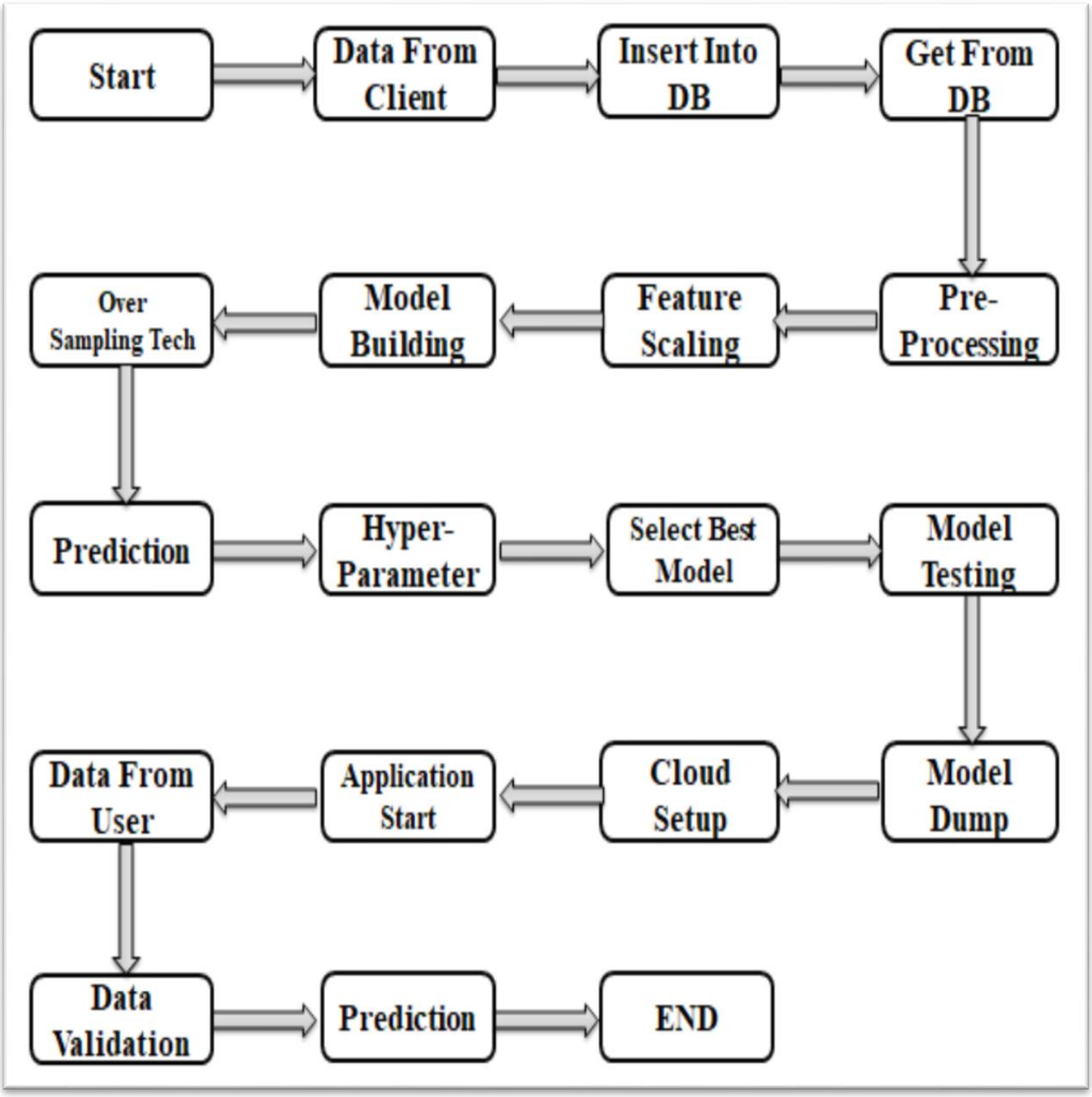

Figure: - The Entire Flow of Machine Learning Flow

3. Architecture Description

3.1 Data Description

This dataset classifies people described by a set of attributes as good or bad credit risks. The data comes in two formats one all numeric & one comes with a cost matrix. The analysis of credit risk depends on the feature that is given in the dataset. There are 20 features available in dataset and one target feature credit risk is present. Total no. of records is 1000 and there is no duplicate value or missing value is present in the dataset. Out of 1000 records 700 records are good risk and 300 records are bad credit risk. The given classification in which the good credit risk is denoted by 1 and bad credit risk is denoted by 0. Two dataset are provided the original dataset, in the form provided by Prof. Hofmann, contains categorical/symbolic attributes and is in the file "German Data". For algorithms that need numerical attributes, Strathclyde University produced the file "German Data-Numeric". This file has been edited and several indicator variables added to make it suitable for algorithms which cannot cope with categorical variables. Several attributes that are ordered categorical (such as attribute 17) have been coded as integer. This was the form used by Stat Log.

1	Laufkont	Status
2	Laufzeit	Duration
3	Moral	Credit_history
4	Verw	Purpose
5	Hoehe	Amount
6	Sparkont	Savings
7	Beszeit	Employment_duration
8	Rate	Installment_rate
9	Famges	Personal_status_sex
10	Buerge	Other_debtors
11	Wohnzeit	Present_residence
12	Verm	Property
13	Alter	Age
14	Weitkred	Other_installment_plans
15	Wohn	Housing
16	Bishkred	Number_credits

17	Beruf	Job
18	Pers	People_liable
19	Telef	Telephone
20	Gastarb	Foreign_worker
21	Kredit	Credit_risk

3.2 Data Insertion into Database

- **Database Creation & Connection -** Create a database with name **South German Bank Data** having keyspace name **credit** and try to create the connection.
- **Create Table** – Check the table inside the keyspace having name with it **credit_data**
- **Insert File** – Insert the data that given by client into the database with help of python script file.
- **Check Data** – Then check that the excel file is uploaded in the dataset or not with the command **SELECT * FROM credit.credit_data.**

3.3 Export from Database

- **Data Export from Database** - The data that we uploaded in database, now we need to pull out from the database for model building.

3.4 Data Pre-Processing

In data pre-processing, we don't do anything special. The reason is that there is no null value or categorical column in the dataset. The name of column is in the German language, so we have to convert it into the English languages. The whole feature names that are in German language & English language are given in data description.

3.5 Feature Scaling

The Feature Scaling technique can be used to scale down the entire feature column in one scale, so that there is not much dispersion in dataset. In our dataset we had used the Standard scalar technique which brings the entire feature column in 0 to 1 range.

3.6 Model Building

After the feature scaling technique we divide the data into train test split format. The train & test data passed to the model that we are using in project i.e. Logistic Regression, Random Forest Classifier, Support Vector Machine, Decision Tree Classifier, XG Boost Classifier, Bagging Classifier, Light GBM Classifier and Gradient Boosting Classifier. Based on the score we select the model for deployment purpose. Before that we need to tune the parameter of each and every module.

3.7 Over Sampling Technique

In that we see the accuracy of model is not that much good. Also we can see that the target feature is not in proper balance. So to balance the target feature we use over sampling technique. In over sampling technique we use ADASYN technique, after that our target feature are equally balance. Then again we train our model with the new target feature and then check the accuracy.

3.8 Hyper Parameter Tuning

In hyper parameter tuning we have implemented ensemble techniques like Random Forest Classifier, XG Boost Classifier, Light GBM Classifier, Bagging Classifier & Gradient Boosting Classifier etc. We have implemented cross validation techniques such as Grid Search CV & Randomized Search CV for different model. From that we have chosen best parameters according to hyper parameter tuning and best score from their accuracies. So we select the Random Forest Regressor as best model from their accuracy 76% as compared to other model.

3.9 Model Testing

After hyper parameter tuning we put all the best parameter in our all ML model. From that we test our data & the score of the model from that we concluded the data score has been increase.

3.10 Model Dump

After comparing all accuracies and checked the score we have chosen hyper parameterized random forest regression as our best model by their results so we have dumped these model in a pickle File format with the help of pickle python module.

3.11 Cloud Setup

After model building we want to deploy the model to server. In deployment we can use different services such as Amazon Web Service (AWS), Azure Service, and Google Cloud Service (GCP). In that we deploy our model in AWS, for that 1st we have to create the environment and then after we have to create the code pipeline. After that we have to connect our pipeline with the environment.

3.12 Data from User

Here we can collect the data from the user. In which we can collect the different type of data such as Status, Duration, Credit_history, Purpose, Amount, Savings, Employment_duration, Installment_rate, Personal_status_sex, Present_residence, Property, Age, Number_credit, and Telephone.

3.13 Data Validation

In data validation the data from user we need to scale it down in between 0 to 1. The reason behind that in our model we scaled down the whole data that’s why we have to scale it down the user entered data.

3.14 Prediction

After entering the data and data validation when user hit the submit button. Internally it will go to app.py file. In that file we have written method called predict it will be executed as you hit the submit button.

4. Technology Stack

1	**Web Framework**	Flask
2	**Database**	Cassandra
3	**Front End**	HTML/CSS
4	**Back End**	Python
5	**Deployment**	Heroku, AWS
6	**Version Control**	GitHub
7	**ML Model**	1] Gradient Boosting Classifier 2] Random Forest Classifier 3] Support vector Machine 4] Decision Tree Classifier 5] Logistic Regression 6] XG Boost Classifier 7] Bagging Classifier 8] Light GBM
8	**IDE**	1] PyCharm 2] VS Code

5. Unit Test Case

Test Case Description	Pre-Requisite	Expected Result
Verify whether the Application URL is accessible to the user	1. Application URL should be defined	Application URL should be accessible to the user
Verify whether the Application loads completely for the user when the URL is accessed	1. Application URL is accessible 2. Application is deployed	The Application should load completely for the user when the URL is accessed
Verify whether user is able to see input fields	1. Application is accessible . User is logged in to the application	User should be able to see input fields
Verify whether user is able to edit all input fields	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be able to edit all input fields
Verify whether user gets submit button to submit the inputs	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should get Submit button to submit the inputs
Verify whether user is presented with recommended results on clicking submit	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	User should be presented with recommended results on clicking submit
Verify whether the recommended results are in accordance to the selections user made	1. Application is accessible 2. User is signed up to the application 3. User is logged in to the application	The recommended results should be in accordance to the selections user made