LOW LEVEL DESIGN (LLD)

Predict Credit Risk Using South German Bank Data

Revision Number : 1.0

Last Date of Revision :10/12/2021

Document Control

Version	Date	Author	Comments
1.0	01/10/2021	Laiju P Joy	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

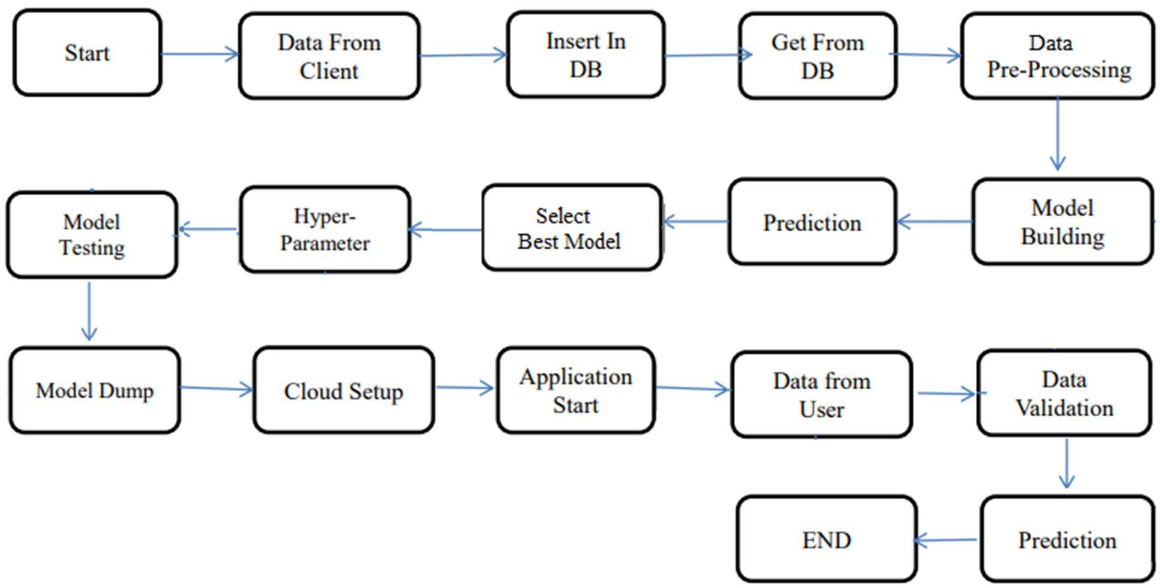

Figure: - The Entire Flow of Machine Learning Flow

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.

	German Words	English Words
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

Attribute Information from German Dataset

Original categorical/symbolic attributes values in all categorical columns of German data described bellow.

Attribute 1: (qualitative)
Status of existing checking account
A11 : ... < 0 DM
A12 : 0 <= ... < 200 DM
A13 : ... >= 200 DM / salary assignments for at least 1 year
A14 : no checking account

Attribute 2: (numerical)
Duration in month

Attribute 3: (qualitative)
Credit history
A30 : no credits taken/ all credits paid back duly
A31 : all credits at this bank paid back duly
A32 : existing credits paid back duly till now
A33 : delay in paying off in the past
A34 : critical account/ other credits existing (not at this bank)

Attribute 4: (qualitative)
Purpose
A40 : car (new)
A41 : car (used)
A42 : furniture/equipment
A43 : radio/television
A44 : domestic appliances
A45 : repairs
A46 : education
A47 : (vacation - does not exist?)
A48 : retraining
A49 : business
A410 : others

Attribute 5: (numerical)
Credit amount

Attibute 6: (qualitative)
Savings account/bonds
A61 : ... < 100 DM
A62 : 100 <= ... < 500 DM

A63 : 500 <= ... < 1000 DM
A64 : .. >= 1000 DM
A65 : unknown/ no savings account

Attribute 7: (qualitative)
Present employment since
A71 : unemployed
A72 : ... < 1 year
A73 : 1 <= ... < 4 years
A74 : 4 <= ... < 7 years
A75 : .. >= 7 years

Attribute 8: (numerical)
Installment rate in percentage of disposable income

Attribute 9: (qualitative)
Personal status and sex
A91 : male : divorced/separated
A92 : female : divorced/separated/married
A93 : male : single
A94 : male : married/widowed
A95 : female : single

Attribute 10: (qualitative)
Other debtors / guarantors
A101 : none
A102 : co-applicant
A103 : guarantor

Attribute 11: (numerical)
Present residence since

Attribute 12: (qualitative)
Property
A121 : real estate
A122 : if not A121 : building society savings agreement/ life insurance
A123 : if not A121/A122 : car or other, not in attribute 6
A124 : unknown / no property

Attribute 13: (numerical)
Age in years

Attribute 14: (qualitative)
Other installment plans
A141 : bank
A142 : stores
A143 : none

Attribute 15: (qualitative)
Housing
A151 : rent
A152 : own
A153 : for free

Attribute 16: (numerical)
Number of existing credits at this bank

Attribute 17: (qualitative)
Job
A171 : unemployed/ unskilled - non-resident
A172 : unskilled - resident

A173 : skilled employee / official
A174 : management/ self-employed/
highly qualified employee/ officer

Attribute 18: (numerical)
Number of people being liable to provide maintenance for

Attribute 19: (qualitative)
Telephone
A191 : none
A192 : yes, registered under the customers name

Attribute 20: (qualitative)
foreign worker
A201 : yes
A202 : no

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,

1. 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.
2. We don't do anything special for missing values, the reason is that there is no null value in the dataset.
3. Drop few non important coloumns in dataset
4. Converting the columns having ordinal values to Label Encoding
5. Converting the columns having non-ordinal values to One Hot Encoding

3.5 Model Building

After the data pre-processing 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, K- Nearest Neighbor and Naïve Bayes Classifier. Based on the score we select the best model for deployment purpose. Before that we need to tune the parameter of selected model.

3.6 Hyper Parameter Tuning

We select the Random Forest Regressor as best model, its **accuracy is 94.3%** , and **F1 score 0.90 for 0** and **0.96 for 1** before hyper parameter tuning.

In hyper parameter tuning, we have implemented Randomized Search CV for model tuning. From that we have chosen best parameters for model according to hyper parameter tuning.

3.7 Model Testing

After hyper parameter tuning we put all the best parameter in our ML model. From that we test our data & the score of the model from that we concluded the data score has been almost same, Accuracy is 94.3% and F1 score 0.90 for 0 and 0.96 for 1.

3.8 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.

3.9 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). Here we deploy our model in AWS

3.10 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.

3.11 Data Validation

In data validation, the data from user we need to validate in correct format or not. Data in correct format only go to Prediction.

3.12 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.Logistic Regression 2.Random Forest Classifier 3.Support vector Machine 4.K- Nearest Neighbor 5.Naïve Bayes
8	**IDE**	1.PyCharm 2.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