Low Level Design (LLD)

Predict Credit Risk Using South German Bank Data

Revision Number : 1.0
Last Date of Revision : 31/08/2024

Document Version Control

Date Issued	Version	Description	Author
31/08/2024	**0.0.1**	**Document Created**	**Kunal Shelke**

1 Introduction

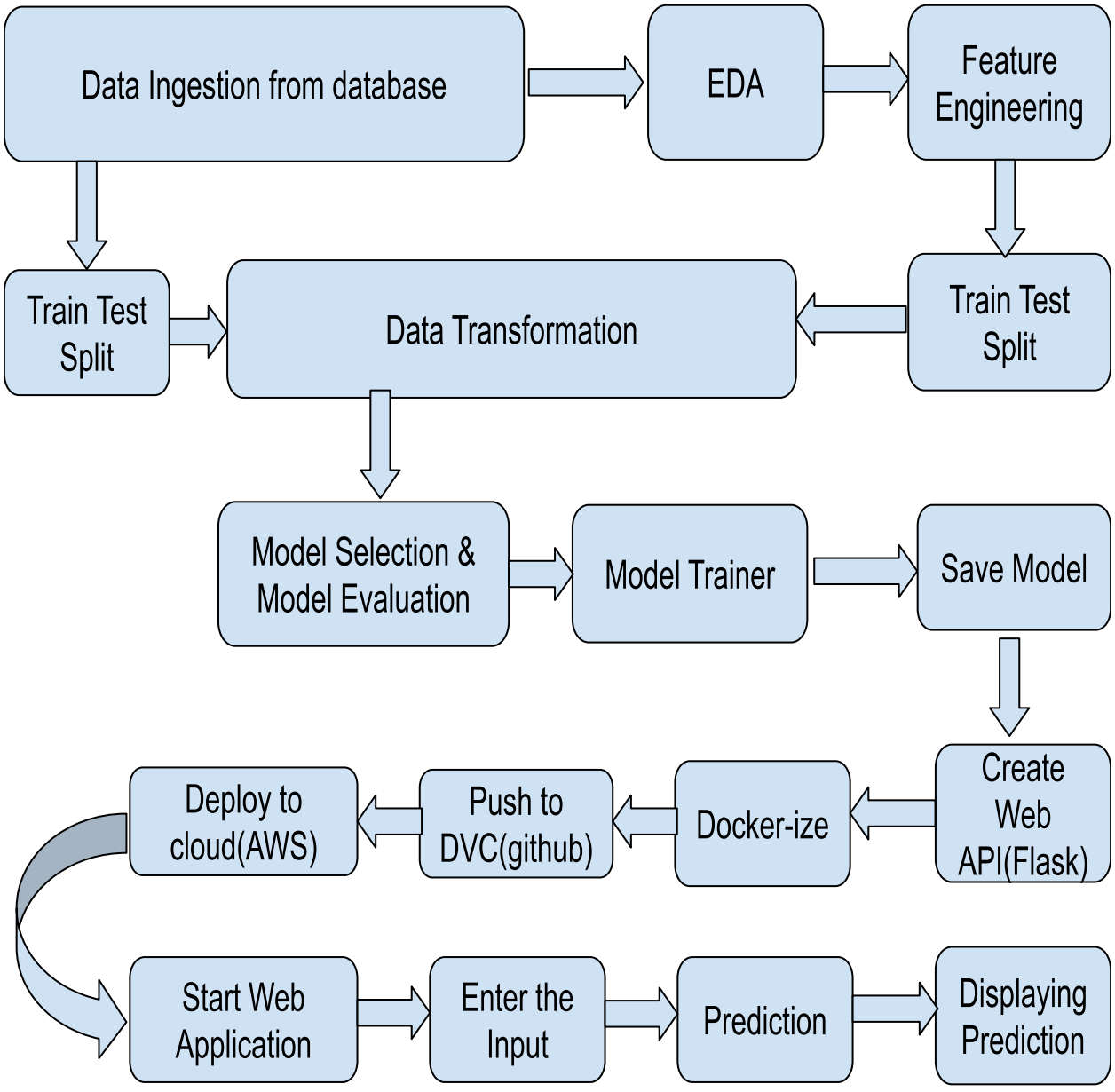
1.1. Why this High-Level Design Document?

The Low-Level Design (LLD) document, also known as the detailed design document, focuses on the implementation details of the system architecture. It breaks down the High-Level Design (HLD) into individual modules and components, specifying the logic, data structures, algorithms, and interactions between different modules. This document is used by developers to code the system and by testers to understand the intricacies of the system's internal workings.

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 Design Details



3. Architecture Description

3.1 Data Description

Here is the detailed description for each column in the dataset:

1. laufkont (status) : Status of the debtor's checking account with the bank.
1: No checking account
2: ... < 0 DM
3: 0 <= ... < 200 DM
4: ... >= 200 DM / salary for at least 1 year

2. laufzeit (duration) : Duration of the credit in months. Numerical value representing the credit duration.

3. moral (credit_history) : History of the debtor's credit at the bank.
0: Delay in paying off in the past
1: Critical account/other credits elsewhere
2: No credits taken/all credits paid back duly
3: Existing credits paid back duly till now
4: All credits at this bank paid back duly

4. verw (purpose) : Purpose for which the credit is being requested.
0: Others
1: Car (new)
2: Car (used)
3: Furniture/equipment
4: Radio/television
5: Domestic appliances
6: Repairs
7: Education
8: Vacation
9: Retraining
10: Business

5. hoehe (amount) : Amount of the credit requested. Numerical value representing the credit amount in DM.

6. sparkont (savings) : Savings account/bonds of the debtor.
   1: Unknown/no savings account
   2: ... < 100 DM
   3: 100 <= ... < 500 DM
   4: 500 <= ... < 1000 DM
   5: ... >= 1000 DM

7. beszeit (employment_duration) : Length of time the debtor has been employed at their current job.
   1: Unemployed
   2: < 1 year
   3: 1 <= ... < 4 years
   4: 4 <= ... < 7 years
   5: >= 7 years

8. rate (installment_rate) : Installment rate as a percentage of disposable income.
   1: >= 35%
   2: 25 <= ... < 35%
   3: 20 <= ... < 25%
   4: < 20%

9. famges (personal_status_sex) : Personal status and sex of the debtor.
   1: Male: divorced/separated
   2: Female: non-single or Male: single
   3: Male: married/widowed
   4: Female: single

10. buerge (other_debtors) : Other debtors or guarantors for the credit.
   1: None
   2: Co-applicant
   3: Guarantor

11. wohnzeit (present_residence) : Length of time the debtor has lived at their current residence.
   1: < 1 year
   2: 1 <= ... < 4 years
   3: 4 <= ... < 7 years
   4: >= 7 years

12. verm (property) : Type of property owned by the debtor.
    1: Unknown/no property
    2: Car or other property
    3: Building society savings agreement/life insurance
    4: Real estate

13. alter (age) : Age of the debtor in years.
    Numerical value representing the age of the debtor.

14. weitkred (other_installment_plans) : Other installment plans held by the debtor.
    1: Bank
    2: Stores
    3: None

15. wohn (housing) : Type of housing the debtor lives in.
    1: For free
    2: Rent
    3: Own

16. bishkred (number_credits) : Number of credits the debtor has in this bank.
    1: 1
    2: 2-3
    3: 4-5
    4: >= 6

17. beruf (job) : Job type of the debtor.
    1: Unemployed/unskilled -non-resident
    2: Unskilled -resident
    3: Skilled employee/official
    4: Manager/self-employed/highly qualified employee

18. pers (people_liable) : Number of people who are financially dependent on the debtor.
    1: 3 or more
    2: 0 to 2

19. telef (telephone) : Whether the debtor has a telephone registered under their name.
    1: No

2: Yes (under customer name)

20. gastarb (foreign_worker) : Whether the debtor is a foreign worker.
    1: Yes
    2: No

21. kredit (credit_risk) : The credit risk assigned to the debtor.
    0: Bad
    1: Good

3.2 Data Ingestion from database

Exporting the data form database that is present in the database that we have to pull out from the database for model building. It can be directly given for data to data transformation for feature scaling

3.3 Data Exploration(EDA)

We divide the data into two types: numerical and categorical. We explore each type one by one. Within each type, we explore, visualize and analyze each variable one by one and note down our observations. Statistical tests are performed for each independent variable to see if the variable has any significance in determining the output for the target variable.

3.4 Feature Engineering

Categorical variables were encoded to make the data suitable for machine learning models. This step involved converting non-numeric data into numeric form

3.5 Train/Test Split

The dataset was split into an 80% training set and a 20% test set, ensuring that the model could be trained on the majority of data while being validated on unseen data.

3.6 Data Transformation

Feature scaling was applied to both the training and test data to normalize the feature distributions, ensuring that the models could perform optimally.

3.7 Model Selection & Model Evaluation

Multiple models were trained and evaluated on the dataset. The best-performing model was identified through comparison of metrics, followed by feature importance analysis and hyper-parameter tuning to enhance model performance.

3.8 Model Trainer

The selected model was then used for training, preparing it for making accurate predictions on new data.

3.9 Save the model

The trained model was saved as a pickle file, making it ready for deployment.

3.10 Create Web Application

A Flask web application was developed, allowing users to input data through a web interface for prediction purposes.

3.11 Dockerised

The application was containerized using Docker, streamlining the process for deployment across different environments.

3.12 Push to DVC(Github)

The project, including the trained model and application files, was version-controlled and pushed to GitHub. The deployment to AWS was automated using GitHub Actions as part of a CI/CD pipeline.

3.13 Deploy to the Cloud

The application was deployed to AWS, enabling it to predict risk classes for unseen data through a cloud-hosted Flask web service

3.14 Application Web Application & Enter the input

Users start the application and enter the required inputs via the web interface.

3.15 Prediction & Display Prediction

Upon submission, the application processes the input data and predicts the risk class, displaying the result as a message indicating whether the customer is classified as Safe or Not Safe..

5. Unit Test Cases

Test Case Description	Prerequisite	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 on logging in	1. Application URL is accessible 2. Application is deployed	User should be able to see input fields on logging in
Verify whether user is able to edit all input fields	1. Application URL is accessible 2. Application is deployed	User should be able to edit all input fields
Verify whether user gets submit button to submit the inputs	1. Application URL is accessible 2. Application is deployed	User should get Submit button to submit the inputs
Verify whether user is presented with recommended results on clicking submit	1. Application URL is accessible 2. Application is deployed	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. Application is deployed	The recommended results should be in accordance to the selections user made
Verify whether user can see its history of records	1. Application is accessible 2. Application is deployed	User should be visible the history of records submitted by them