LOW LEVEL DESIGN DOCUMENT

Prediction of LC50 Value Using Quantitative Structure Activity Relationship (QSAR) Models

Written By	Rizal Muhammed
Document Version	1.0
Last Revised Date	20/03/2024

Document Control

Change Record

Version	Date	Author	Comments
1.0	20 / 03 / 2024	Rizal Muhammed	Introduction & Architecture defined

Reviews

Version	Date	Reviewer	Comments

Approval Status

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1 Introduction

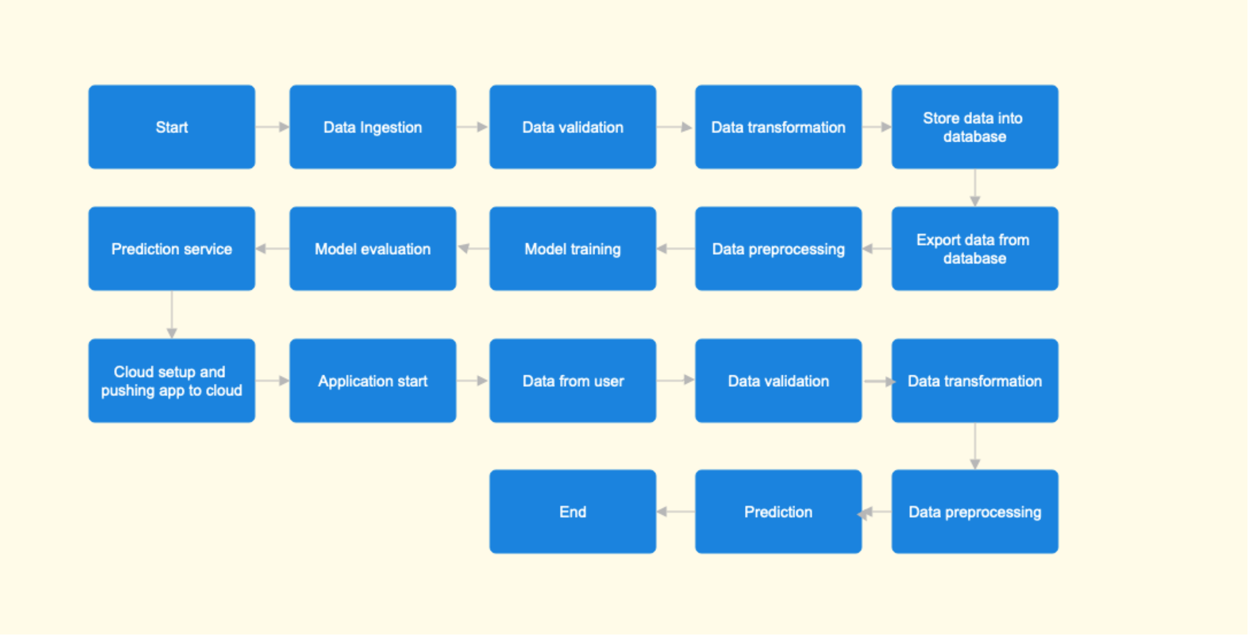
1.1 What is a Low-Level design document?

The goal of LLD (Low-level design document) is to give the internal logical design of the actual program code for Prediction of LC50 value using QSAR models. 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 document 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



3 Architecture Description

3.1 Data Description

QSAR fish toxicity data set is available at UCI Machine Learning Repository. Data set containing values for 6 attributes (molecular descriptors) of 908 chemicals used to predict quantitative acute aquatic toxicity towards the fish Pimephales promelas (fathead minnow). Please refer data set here.

3.2 Data Ingestion

Create folder artifacts/data_ingestion and download, extract data set into data directory. So, at the end of data ingestion stage, the CSV data, qsar_fish_toxicity.csv will be available in artifacts/data_ingestion/data directory.

3.3 Data Validation

Perform data validation such as raw file name validation, column length validation, validating missing values in whole columns, etc. If the data fails data validation, the corresponding files are moved to artifacts/data_validation/archive_bad_dir directory for reviewing. Upon successful data validation, the data is copied into artifacts/data_validation/tranining_raw_files_validated directory.

3.4 Data Transformation

Perform data transformation like replacing missing value with NULL. Remove existing good data directory and bad data directory. Move bad data from bad directory to archive_bad_directory.

3.5 Data Base Operations

Create table in the database (if it does not exist), insert good data into database table, Export data from table into csv file. At the end of this stage, the input file will be available at artifacts/tranining_file_from_db/inputfile.csv directory.

3.6 Data Preprocessing

In this stage, load the input data for training. Separate data into labels and features. Perform imputing missing values. Eliminate columns with zero standard deviation. Perform standard scaling and save data. At the end of this stage, the features data should be available at artifacts/preprocessed_data/preproccessed_input_X and the labels data should be available at artifacts/preprocessed_data/preprocessed_input_Y

3.7 Model Training

In this stage, model building and hyper parameter tuning should be performed. Different machine learning algorithms are tried w.r.t dataset and save the best model along with corresponding performance metrics from the best model

3.8 Prediction Service

After Model training and evaluation, a prediction service is deployed in the cloud and made available to the user. The user can enter data into the front end of the prediction service. The best model will be loaded and should predict the output based on the input user data

3.9 Model Deployment

We will be deploying the model to Heroku

4 Unit Test Cases

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	Application should load completely for the user when the URL is accessed
Verify whether user can see input fields on accessing the URL	1. Application is accessible	User should be able to successfully see input fields
Verify whether user can edit all input fields	1. Application is accessible 2. All the input fields are visible	User should be able to edit all input fields
Verify whether user gets Predict button to submit the inputs	1. Application is accessible 2. All the input fields are visible 3. User can edit input fields	User should be able access Predict button
Verify whether the corresponding prediction is displayed	1. Application is accessible 2. All the input fields are visible 3. User can edit input fields 4. User can submit input	User should be able to get the prediction displayed