Low Level Design (LLD)

Concrete Compressive strength Prediction

|  |  |
| --- | --- |
| Written by | Sree Harsha K |
| Document version | 0.2 |
| Last revised date | 28 /11/2022 |

# Document Version Control

## Change record:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
| 0.1 | 27-Sep-21 | Sree Hrahs K | Introduction and architecture defined |
| 0.2 | 28-Oct-21 | Sree harsha K | Architecture and architecture  description updated. Unit test cases defined and updated |
|  |  |  |  |
|  |  |  |  |

**Reviews:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Reviewer** | **Comments** |
| 0.2 |  |  |  |

## Approval status:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Review Date** | **Reviewed By** | **Approved By** | **Comments** |
|  |  |  |  |  |

Contents

1. [Introduction 4](#_TOC_250012)
   1. [What is Low-Level design document? 4](#_TOC_250011)
   2. [Scope 4](#_TOC_250010)
2. [Architecture 5](#_TOC_250009)
3. [Architecture Description 6](#_TOC_250008)
   1. [Data Collection 6](#_TOC_250007)
   2. [Data ingestion 7](#_TOC_250006)
   3. [Data pre-processing 7](#_TOC_250005)
   4. [Model building 8](#_TOC_250004)
   5. [Model evaluation 10](#_TOC_250003)
   6. [Saving the best models 11](#_TOC_250002)
   7. [Model deployment 11](#_TOC_250001)
4. [Unit Test Cases 12](#_TOC_250000)

# Introduction

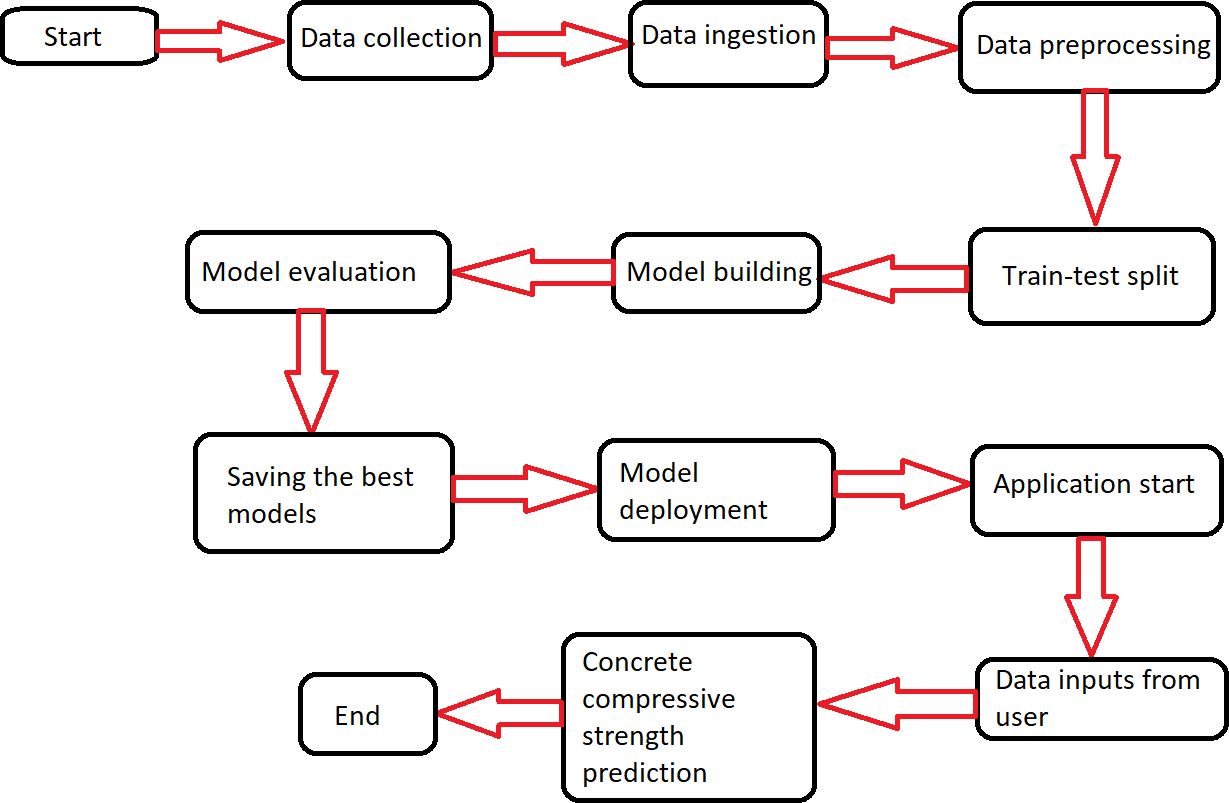
## What is Low-Level design document?

The goal of LLD or a Low-level design document is to give an internal logical design of the actual program code for the Concrete Compressive Strength Prediction 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.

## 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 defined during data design work.

# Architecture



# Architecture Description

## Data Collection

For training and testing the model, we used the public data set available in Kaggle,

“Concrete Compressive Strength Data Set” by Ahiale Darlington.

URL - <https://www.kaggle.com/elikplim/concrete-compressive-strength-data-set>

Data dictionary is as follows:

The actual concrete compressive strength (MPa) i.e., mega pascals for a given mixture under a specific age (days) was determined from laboratory. Data is in raw form (not scaled).

Summary Statistics:

Number of instances (observations): 1030 Number of Attributes: 9 Attribute breakdown: 8 quantitative input variables, and 1 quantitative output variable. Missing Attribute Values: None

Variable Information:

Given is the variable name, variable type, the measurement unit and a brief description. The concrete compressive strength is the regression problem. The order of this listing corresponds to the order of numerals along the rows of the database.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data Type | Measurement | Description |
| Cement | Quantitative | kg in a m3 mixture | Input variable |
| Blast Furnace Slag | Quantitative | kg in a m3 mixture | Input variable |
| Fly Ash | Quantitative | kg in a m3 mixture | Input variable |
| Water | Quantitative | kg in a m3 mixture | Input variable |
| Superplasticizer | Quantitative | kg in a m3 mixture | Input variable |
| Coarse Aggregate | Quantitative | kg in a m3 mixture | Input variable |
| Fine Aggregate | Quantitative | kg in a m3 mixture | Input variable |
| Age | Quantitative | Days (1~365) | Input variable |
| Concrete Compressive  Strength | Quantitative | megapascals (MPa) | Output variable |

## 3.2 Data pre-processing

## Described the data to know the descriptive statistics summary between the features in data.

## Getting the info of data to know the datatype and null values in data.

## Plotting the boxplot by using seaborn to know the outliers of the data.

## Plotting the scatterplot to know the normally distribution data.

## Dividing and Training the data into feature variables and target value.

## 3.3 Model building

## 

## Linear Regression-

## r 2\_score in train model- 55%

## r 2\_score in test model-63%

## Random forest Regressor-

## r 2\_score in train model- 87%

## r 2\_score in test model-98%

## Ada Boost Regressor

## r 2\_score in train model- 78%

## r 2\_score in test model-82%

## Xgboost

## r 2\_score in train model- 99.9%

## r 2\_score in test model-89.99%

## Grid Search CV is used to find the best parameters of the Xgboost

## r 2\_score in train model- 99%

## r 2\_score in test model-92%

## 

## 3.4 Saving the best models

Based on the above results, I saved the “XGBoost regressor” model directory using the joblib library.

## Model deployment

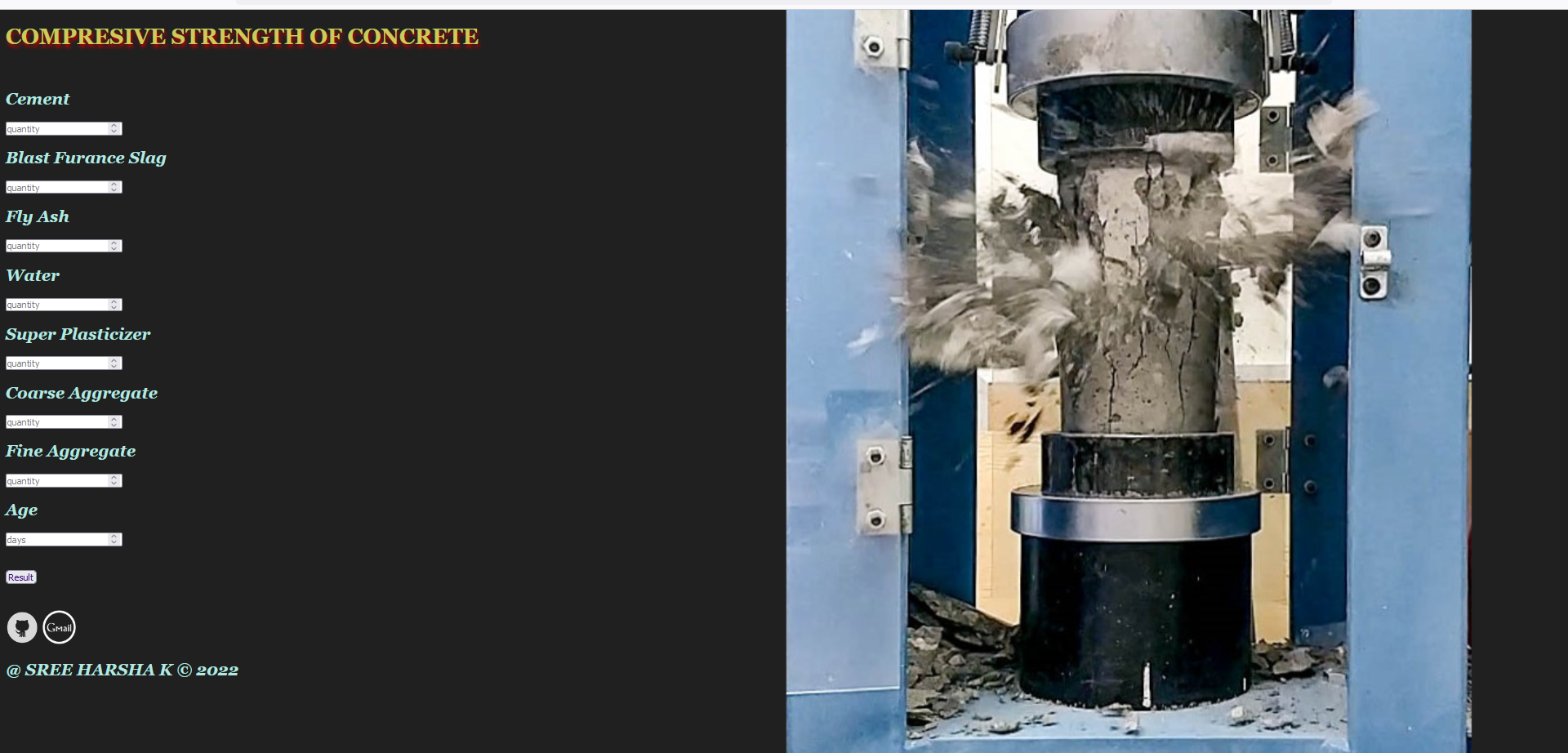
Deployed the “XGBoost regressor” model in the web application using Flask a micro web framework in python. The deployment part of the code runs in the “app.py” file, connecting with the web page designed using HTML with CSS styles.

Then, deployed on web using the GitHub a version control system, Gunicorn version

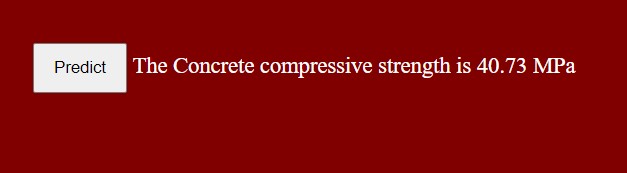
20.1.0 and the Heroku a platform as a service (PaaS).

Web application URL -<https://compressivestrength.herokuapp.com/>

Here’s how the web application looks like:



Users or engineers can input the details about the concrete specimen and once they hit “Predict” button, the predicted concrete compressive strength will be displayed as below



# 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 | The application should load completely for the user when the URL is accessed |
| Verify whether user can edit all the input fields | 1. Application URL is accessible 2. Application loads completely for the user. 3. All the input fields   loaded | User should be able to edit all the input fields |
| Verify whether user gets “Predict” button to make predictions on the given inputs | 1. Application URL is accessible 2. Application loads completely for the user. 3. All the input fields loaded | User should get a “Predict” button to make predictions on the given inputs. |
| Verify whether user is presented with recommended results on clicking the “Predict” button | 1. Application URL is accessible 2. Application loads completely for the user 3. All the input fields   loaded | Users should be presented with recommended results on clicking the “Predict” button. |