**Low-Level Design (LLD)**

**Concrete Compressive Strength Prediction**

|  |  |
| --- | --- |
| Written By | SHREYAS KHADYE  SUDHANSU GOUDA |
| Version | 1.0 |
| Date | 01/04/2023 |

**Document Change Control Record**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comments** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Reviews:

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

### Approval Status:

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

**Index**

|  |  |
| --- | --- |
| **Content** |  |
| 1. Introduction | 4 |
| 1.1 What is Low-Level Design Document | 4 |
| 1.2 Scope | 4 |
| Architecture | 5 |
| 2. Architecture Description | 5 |
| 2.1 Data Description | 5 |
| 2.2 Data Gathering | 5 |
| 2.3 Raw Data Validation | 5 |
| 2.4 Data Transformation | 6 |
| 2.5 New Feature Generation | 6 |
| 2.6 Data Pre-processing | 6 |
| 2.7 Model Building | 6 |
| 2.8 Model Saving | 6 |
| 2.9 Web app setup | 6 |
| 2.10 GitHub | 6 |
| 2.11 Deployment | 7 |
| 3. Unit Test Cases | 7 |

**1. Introduction**

**1.1 What is Low-Level Design Document.**

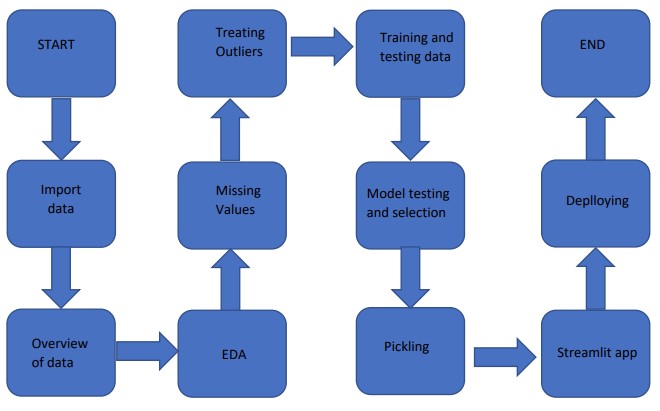
The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for **‘Concrete Compressive Strength Prediction’**. LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that 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.

**Architecture**

Process Flow

[](https://user-images.githubusercontent.com/123532199/228265721-f4f5c0d3-5437-4576-bdb5-c8e3d2889a7f.jpg)

**2. Architecture Description**

**2.1 Data Description**

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** |
| cement | Integer | kg in a m3 mixture |
| Blast Furnace Slag | Integer | kg in a m3 mixture |
| Fly Ash | Integer | kg in a m3 mixture |
| Water | Integer | kg in a m3 mixture |
| Superplasticizer | Integer | kg in a m3 mixture |
| Coarse Aggregate | Integer | kg in a m3 mixture |
| Fine Aggregate | Integer | kg in a m3 mixture |
| Age | Integer | Day (1~365) |
| Concrete compressive strength | Integer | MPa |

**2.2 Data Gathering**

Data source:

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

**2.3 Raw Data Validation**

After data is loaded, various types of validation are required. Validations like checking for zero standard deviation for all the columns, checking for complete missing values in any columns, etc. These are required because The attributes which contain these are of no use. It will not be much of use in determining of the salary category. Like if any attribute is having zero standard deviation, it means that’s all the values are the same, its mean is zero. Missing data in the training data stands as a problem in front of us.

**2.4 Data Transformation**

Before sending the data into the database, data transformation is required so that data are converted into such a form with which it can easily be inserted into the database. Here, the ‘blast\_ furnance\_slag’ ,‘fly\_ Ash’ and ‘superplasticizer’ attributes contain a vast amount of missing values. So, they have been replaced them with the respective mean values.

**2.5 New feature generations**

We haven’t derived a new category.

**2.6 Data Pre-processing**

In data pre-processing all the processes required before sending the data for model building are performed. Like, here ‘blast\_ furnance\_slag’ ,‘fly\_ Ash’ and ‘superplasticizer’ the attributes are having some values equal to 0,no doubt both of these attributes are viable for prediction due to maximum entry being zero in the model they can’t contribute much. So they have been replaced with the respective mean values. We also detected some outliers with the help of bar chart and replaced them with the respective mean values.

**2.7 Model building:**

After doing all kinds of preprocessing operations mention above and performing model training and testing the accuracy we came to the conclusion that Lasso Regression have the highest accuracy with 93% accuracy.

**2.8 Model saving:**

Model is saved using the pickle library in ‘. pkl’ format.

**2.9 Web App Setup:**

After saving the model in .pkl file format we then create an app.py streamlit web app framework (Written in python) and then use requests to extract all the form selection selected by the user and then we predict the salary prediction by using the selected records by the user.

**2.10 Git Hub:**

The whole project directory will be pushed into the GitHub repository.

GitHub Project link:

https://github.com/ShreyasKhadye/Concrete-Compressive-Strength-Prediction

**2.11 Deployment:**

The cloud environment was set up and the project was deployed from GitHub into the Streamlit Sharing cloud platform.

WebApp link –

[https://shreyaskhadye-concrete-compressive-strength-predicti-app-l56p7u.streamlit.app/](%20https://shreyaskhadye-concrete-compressive-strength-predicti-app-l56p7u.streamlit.app/)

**3. 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 a user is able to see input fields while opening the application | 1. Application is accessible 2. The user is able to see the input fields | Users should be able to see input fields on logging in |
| Verify whether a user is able to enter the input values. | 1. Application is accessible 2. The user is able to see the input fields | The user should be able to fill the input field |
| Verify whether a user gets predict button to submit the inputs | 1. Application is accessible 2. The user is able to see the input fields | Users should get Submit button to submit the inputs |
| Verify whether a user is presented with recommended results on clicking submit | 1. Application is   accessible   1. The user is able to see the input fields. 2. The user is able to see the submit button | Users should be presented with recommended results on clicking submit |
| Verify whether a result is in accordance with the input that the user has entered | 1. Application is accessible 2. The user is able to see the input fields. 3. The user is able to see the submit button | The result should be in accordance with the input that the user has entered |