Concrete Strength Prediction

Machine learning project

Data 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.

The actual concrete compressive strength (MPa) for a given mixture under a specific age (days) was determined from laboratory.

Input Features

- Cement (cement)
- Blast Furnace Slag
- Fly Ash
- Water (water)
- Super plasticizer (super_plastic)
- Coarse Aggregate (coarse agg)
- Fine Aggregate (fine agg)
- Age(age) -- quantitative

Target Feature

- Concrete compressive strength(strength)

Data Source :https://www.kaggle.com/datasets/elikplim/concrete-compressive-strength-data-set

Sources:

Original Owner and Donor

Prof. I-Cheng Yeh

Department of Information Management

Chung-Hua University,

Hsin Chu, Taiwan 30067, R.O.C.

e-mail:icyeh@chu.edu.tw

TEL:886-3-5186511

Date Donated: August 3, 2007

Problem Description

Concrete, as the cornerstone of civil engineering, holds paramount importance. Its compressive strength, influenced by age and various ingredients, follows a highly nonlinear pattern. The key ingredients encompass cement, blast furnace slag, fly ash, water, super plasticizer, coarse aggregate, and fine aggregate.

Our objective lies in constructing a machine learning model utilizing the existing data to predict the compressive strength of concrete component.

Tools Used

- ✓ Python 3.8 is used while creating the environment and libraries like NumPy, Pandas, Scikit-learn.
- ✓ Pymongo, Exceptions and logger are used for developing the model.
- ✓ VS code is used to development the modular model
- ✓ MongoDB is used to store and retrieve the data.

Data Base

The MongoDB database plays a crucial role in the training pipeline by facilitating the acquisition of data files for the training process. Additionally, it serves as a convenient platform for uploading prediction data when required.

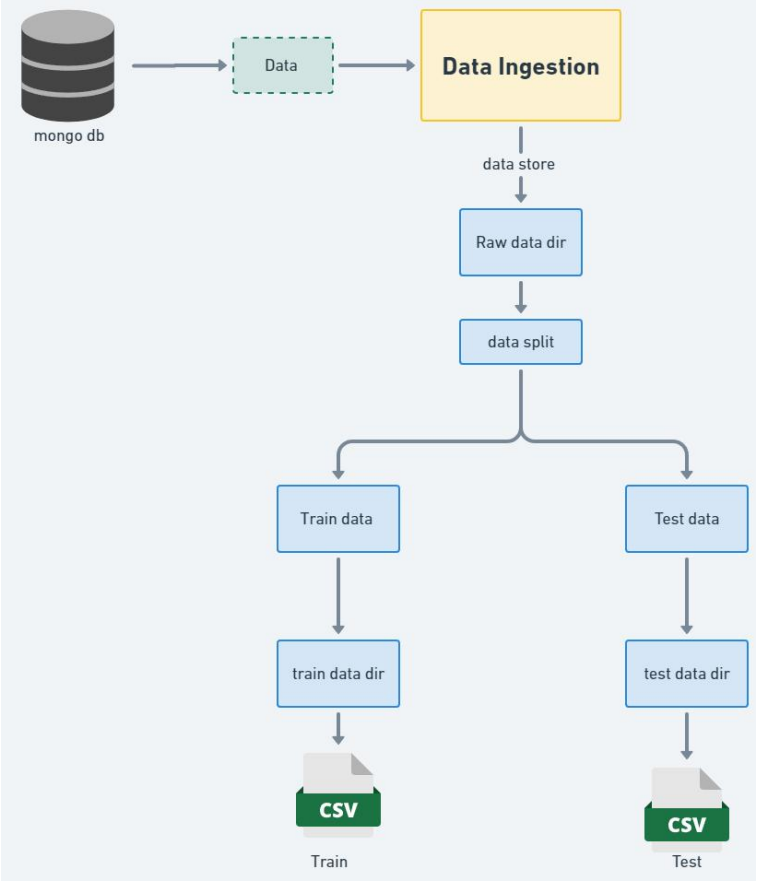
Mongo Database

Training Pipeline

Components of Pipeline

- Data Ingestion
- Data Validation
- Data Transformation
- Model Training
- Model Evaluation
- Model Pusher

Data Ingestion



Data Ingestion class represents a data ingestion process that involves retrieving data from a data source, saving it in a raw data directory, and splitting it into training and test datasets.

The class code flows in following way:

Data Ingestion class takes a necessary elements and that provides the configuration settings for the data ingestion process.

It retrieves data from a data source and saves it in a raw data directory. It accesses the configuration settings to determine the location of the raw data directory and performs the necessary operations to obtain the data from the data source and saves it in the specified directory.

This method splits the raw data into training and test datasets. It reads the data from the raw data directory and split it into two sets:

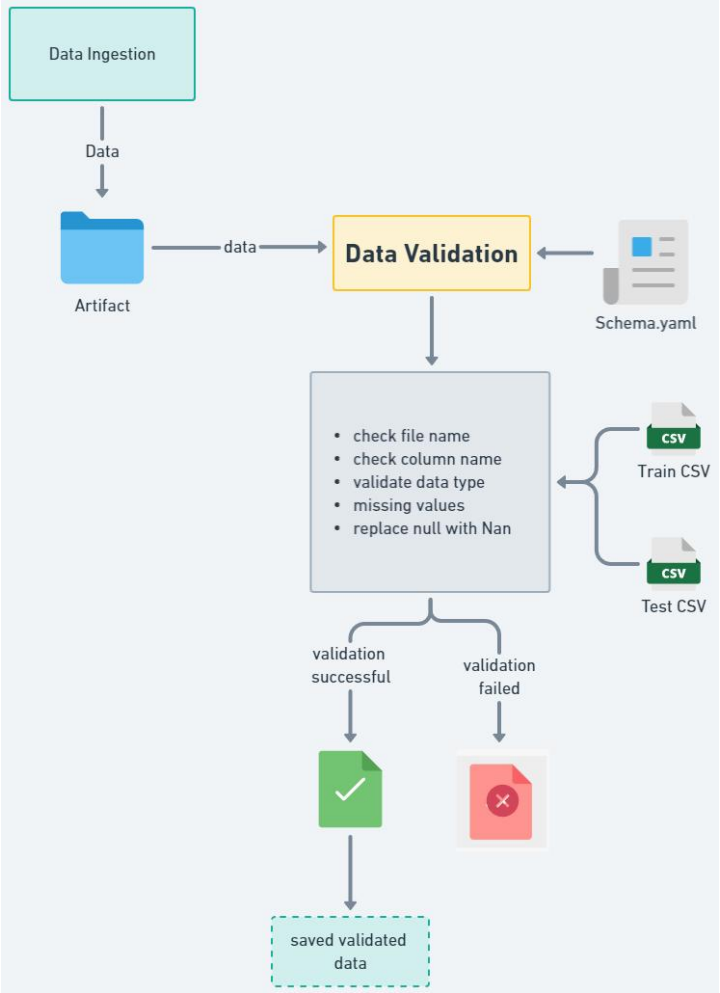
a. training set

b. test set.

The method determines the split ratio based on the configuration settings.

Throughout the process, the class handles any exceptions that may occur and raises an application exception with an appropriate error message.

Data Validation



This method aims to validate the training and test data stored in the artifact folder as a result of the data ingestion process carried out earlier. Its responsibility is to perform validation checks on the training and test datasets.

The method follows a series of steps to validate the data and logs the validation process. If the validation is successful, it exports the validated datasets to specified paths and returns the paths. However, if the validation fails, it raises an exception.

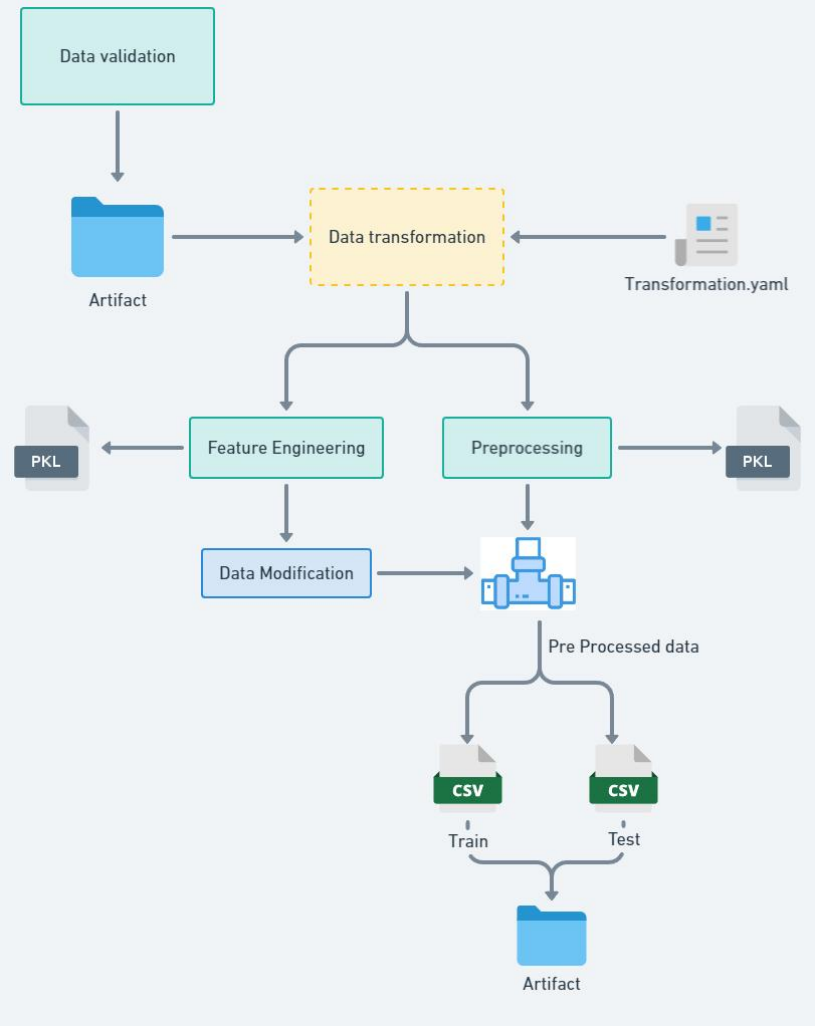
Validation processes :

- File name of the downloaded dataset
- Column labels
- Validating Data types
- Missing values whole column
- Replace null values

Each of these methods returns a boolean value as the result of the validation. If the dataset passes all the checks and evaluates to True, it is stored in the artifact folder.

Throughout the process, the class handles any exceptions that may occur and raises an application exception with an appropriate error message.

Data Transformation



Once the validated data is obtained from the artifact folder, it proceeds for the necessary transformations before being used for model training. These transformations ensure that the data is in a suitable format and structure for the training process.

Feature Engineering

The feature engineering class leverages insights from exploratory data analysis (EDA) to transform the dataset, enriching it with new and meaningful information. These engineered features have the potential to improve the model's performance and predictive capacity.

To ensure consistent utilization of these features, a feature engineering object file is created, allowing for batch prediction on new data without repeating the entire feature engineering process. This approach promotes consistency, reproducible, and efficient application of pre_processing steps to the data.

Pre_processing Pipeline

After feature engineering the data, the next step is pre_processing . The pre_processing pipeline applies a series of steps to the training and testing datasets, ensuring they are in a consistent and suitable format for model training.

The pre_processing pipeline plays a critical role in optimizing the data for model training by standardizing and cleaning it. This enables the model to learn patterns effectively during training and make accurate predictions.

Overall, the data transformation stage ensures that the validated data undergoes necessary transformations, while feature engineering enhances the dataset by creating new features. The pre-processing pipeline then applies a series of standardized steps to prepare the data for model training.

Output :

Object Files

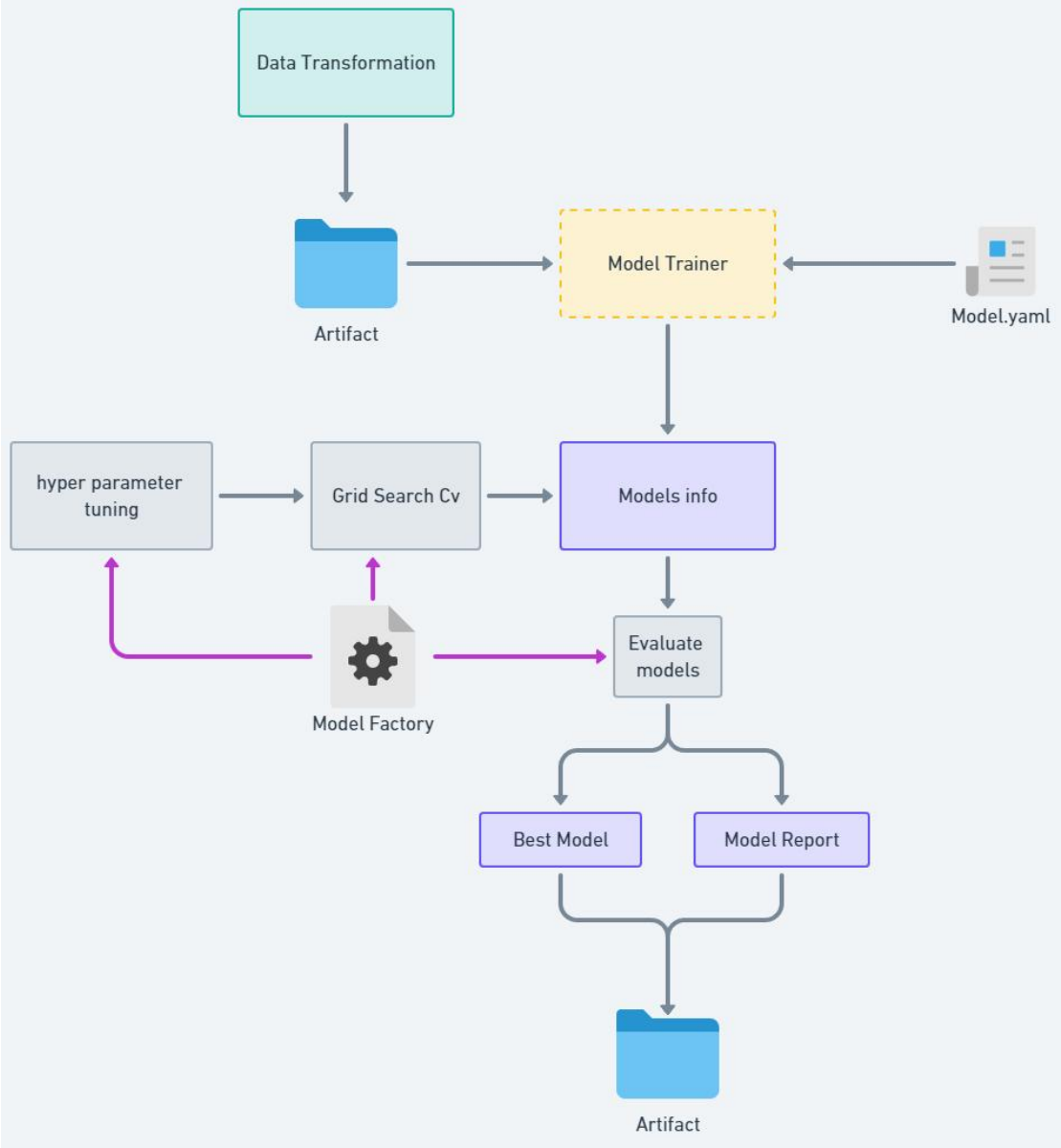
Feature Engineering Pipeline ---> pkl object

Pre_processing pipeline ---> pkl object

Transformed Data

- Train Data
- Test Data

Model Training



Transformed train and test datasets is further used for model training

Models:

For this regression problem statement based on the Data analysis we detect few relevant models for model training :

- Random forest
- XG Boost
- Ada Boost

Hyper_parameter Tuning and Grid Search CV:

Hyper_parameter tuning involves finding the optimal combination of hyper parameters for a given model.

Grid Search Cross-Validation (Grid Search CV) is a technique that exhaustively searches through a specified hyper_parameter grid to find the best parameter values for the models. It systematically evaluates different hyper parameter combinations using cross-validation, ensuring the model's performance is robust .

Metric used : R_2 score

After training the models, the R_2 score is calculated for both the train and test data sets. The difference in the R_2 scores between the train and test data for each model is examined.

If the difference is less than 0.20 (indicating a small difference), the model is considered suitable for further analysis.

Output :

Model Object

The best model, along with its corresponding best parameters, is saved for future use. This ensures that the optimal model can be easily retrieved and applied to new data.

Model Report

Report containing relevant information and insights is also saved. This report provides a concise summary of the model's performance and any additional context or analysis that may be necessary for understanding the model's results.

Model object and report is saved to the artifact directory

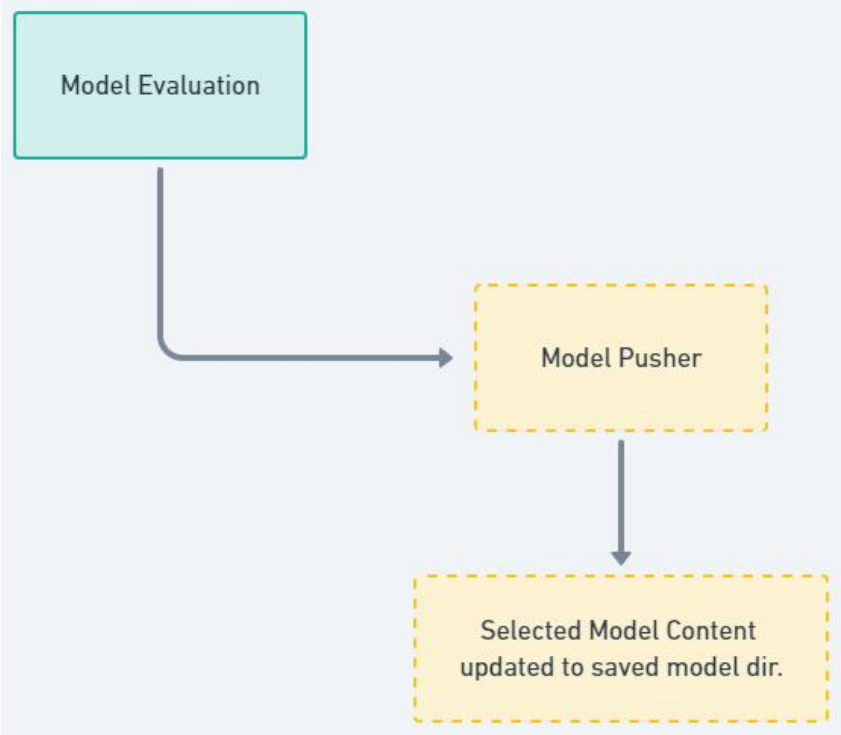
Model Evaluation

The saved model directory may contain earlier models and other necessary components from previous iterations.

During the evaluation process, we compare the recently trained artifact model with the previously saved models. By leveraging this comparison, we identify the best-performing model.

Once the selection is made, we save the chosen model along with its associated model object and a comprehensive model report. This approach ensures that we retain the most optimal model for future use.

Model Pusher



Upon selecting the models and gathering the relevant information from the evaluation module, we proceed to push the chosen model and its corresponding report to the saved model directory. This ensures that the selected model is readily available for future prediction processes.

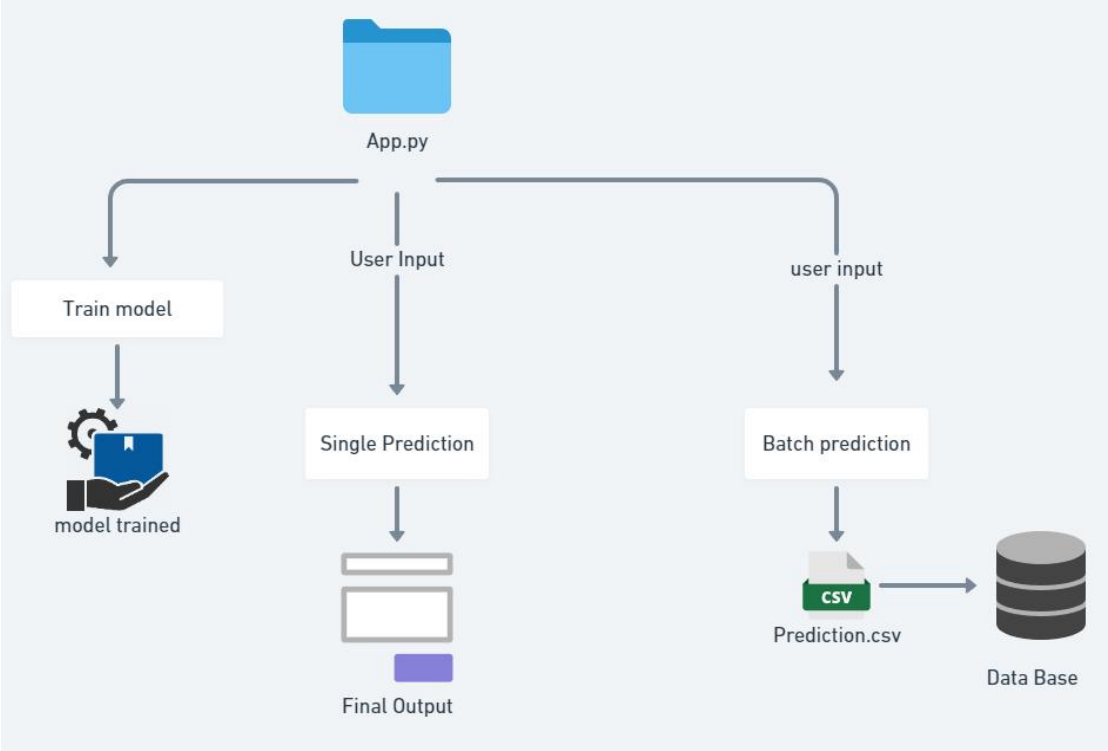
Saved Model Directory

Model

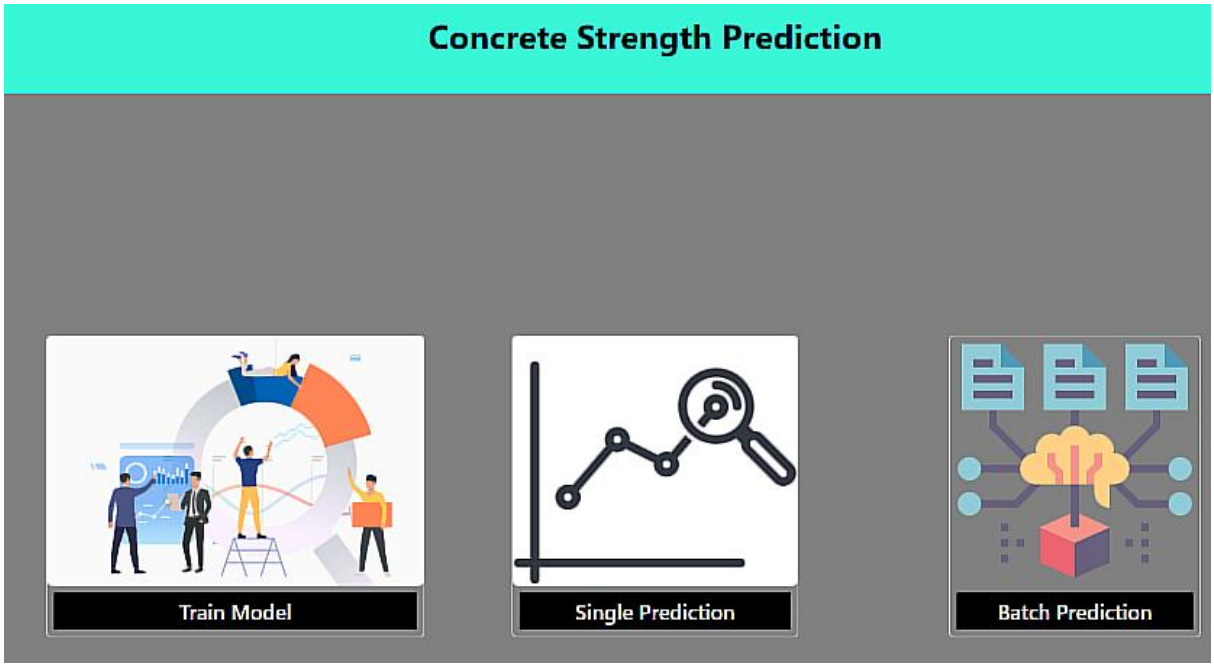
Model Report

By storing it in the designated directory, we facilitate easy access and utilization of the model in subsequent tasks.

Application



UI



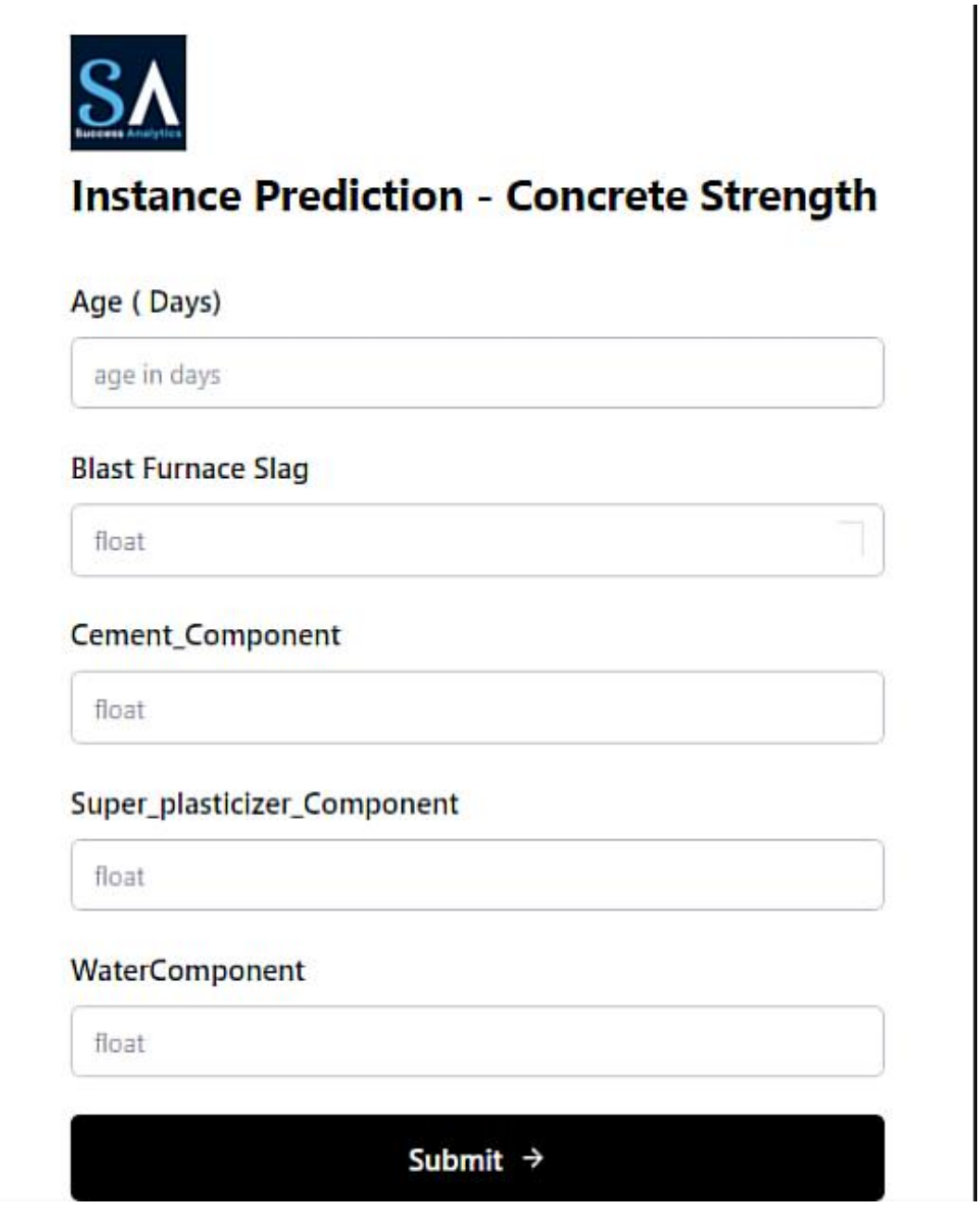
Train Model

This option triggers an entire training pipeline to train the model

Instance Prediction

After training the model and saving the necessary contents and information, the system utilizes the user's input to generate predicted results.

These results are then displayed on an HTML web page for easy access and visualization



Batch Prediction

In batch prediction, the user provides a CSV file as input, and the system generates a prediction CSV file. The resulting file is then stored at a specified location and can also be uploaded to a MongoDB database for further use or analysis.