Detail Project Report (DPR)

Concrete Strength Prediction

Revision Number : 1.1
Last Date of Revision: 16/07/2023

Document Version Control

Date Issued	Version	Description	Author
15-07-2023	1.0	Initial DPR - V1.0	Aniket Lotlikar
16-07-2023	1.1	Updated DPR – V1.1	Aniket Lotlikar

Contents

Abstract

Being one of the most frequently used building materials, the quality of concrete is determined by its compressive strength, which is measured by crushing a concrete cube or a cylinder until it starts cracking and crushing. The pressure at which the concrete cube or a cylinder starts cracking and eventually crushes is called the Concrete compressive strength and is measured in megapascals (MPa). It takes a long period of 28 days to test like this. With the help of Data Science and Machine learning technology, I developed an application, which allows an engineer to determine the strength of a concrete in just a few seconds of time.

1. Introduction

1.1 Why this DPR Document?

The main purpose of this DPR documentation is to add the necessary details of the project and provide the description of the machine learning model and the written code. This also provides the detailed description on how the entire project has been designed end-to-end. Key points:

- Describes the design flow
- Implementations
- Software requirements
- Architecture of the project
- Non-functional attributes like:
- Reusability
- Portability
- Resource utilization

2. General Description

2.1 Problem Perspective

The Concrete Strength Prediction is a machine learning model that helps users to understand their Concrete Compressive Strength based on some input data.

2.2 Problem Statement

The main goal of this model is to predict Insurance premium price based on some input data like cement , blast furnace , fly ash etc.

2.3 Proposed Solution

To solve the problem, we have created a user interface for taking the input from the user to predict insurance premium price using our trained ML model after processing the input and at last the predicted value from the model is communicated to the user

3. Technical Requirements

As technical requirements, we don't need any specialized hardware for virtualization of the application. The user should have a device that has the access to the web and the fundamental understanding of providing the input. And for the backend, we need a server to run all the required packages to process the input and predict the desired output.

3.1 Tools Used

Python programming language and frameworks such as NumPy, Pandas, Scikit-learn, Flask, VS Code and are used to build the whole model.
- VS Code is used as an IDE.
- For visualization of the plots, Matplotlib, Seaborn and Plotly are used.
- Azure Cloud is used for deployment of the model.
- Front end development is done using HTML,CSS and Bootstrap Framework .
- Python Flask is used for backend development.
- GitHub is used as a version control system.

4. Data Requirements

The Data requirements totally supported the matter statement and also the dataset is accessible on the Kaggle within the file format of (.zip).

4.1 Data Collection

The Data requirements totally supported the matter statement and also the dataset is accessible on the Kaggle within the file format of (.zip). The data for this project is collected from the Kaggle Dataset, the URL for the dataset is :-
https://www.kaggle.com/elikplim/concrete-compressive-strength-data-set

4.2 Data Description

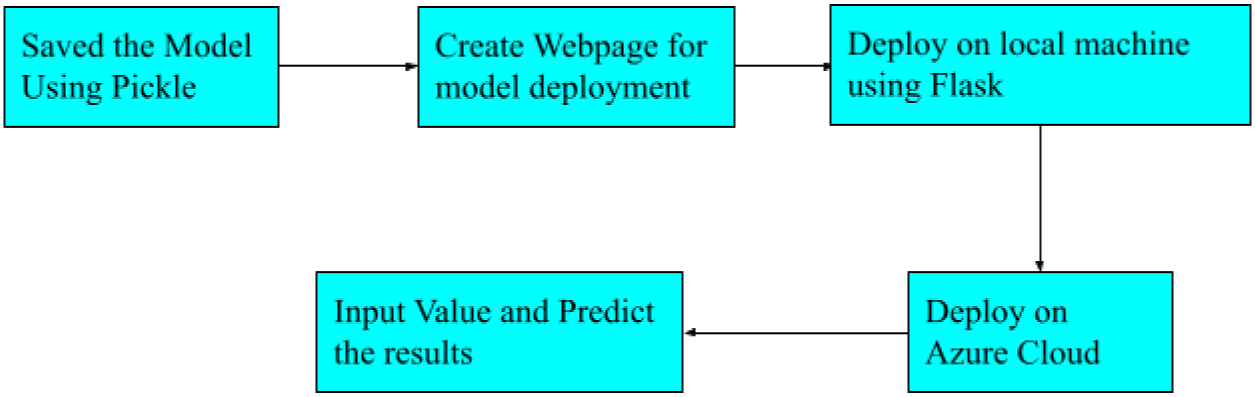
The Concrete Strength Prediction dataset publicly available on Kaggle. The information in the dataset is present in one csv file named concrete_data.csv. Dataset contains 1030 rows * 9 columns.

4.3. Data Pre-processing

In logging, at each time an error or an exception occurs, the event is logged into the system log file with reason and timestamp. This helps the developer to debug the system bugs and rectify the error.
- Checked for info of the Dataset, to verify the correct datatype of the Columns.
- Checked for Null values and also the duplicated values , because the null values can affect the accuracy of the model.
- Checking the distribution of the columns to interpret its importance. Now, the info is prepared to train a Machine Learning Model

5. Design Flow

5.1 Deployment Process


Saved the Model Using Pickle
Create Webpage for model deployment
Deploy on local machine using Flask
Deploy on Azure Cloud
Input Value and Predict the results

5.2 Logging

In logging, at each time an error or an exception occurs, the event is logged into the system log file with reason and timestamp. This helps the developer to debug the system bugs and rectify the error.

5.3 Data from User

The data from the user is retrieved from the web page.

5.4 Data Validation

The data provided by the user is then being processed by app.py file and validated.

The validated data is then sent to the prepared model for the prediction

5.5 Rendering the Results

The data sent for the prediction is then rendered to the web page

6. Deployment

The tested model is then deployed to Flask . So, users can access the project from any internet device.

7. Conclusion

The Concrete Compressive Strength system will predict the price for helping the customers with the trained knowledge with a set of rules. The user can use this system to recognize the approximate value of their Concrete Compressive Strength

8. Frequently Asked Questions (FAQs)

1) What's the source of data?

Answer: The data for training the model is taken from Kaggle - https://www.kaggle.com/elikplim/concrete-compressive-strength-data-set

2) What's the complete flow you followed in this project?

Answer: Please refer to slide 3 for a better understanding.

3) What techniques were you using for data-preprocessing?

Answer: 1. Visualizing the relationships between the target variable and each predictor variable.
2. Visualizing the correlation among all the variables.
3. Removing outliers using IQR method from the column and increasing the size of the data
4. Scaling the data for building Linear regression models using Standard scaler.

4) What's the purpose of increasing the size of data?

Answer: Having less amount of data makes the model to learn completely on it (in addition to learning the underlying patterns, it also learns the noise in that data), leading to overfit. Means, the model may not make predictions accurately on the unseen data. So, to avoid model overfit, one of the methods is to add more data points so that the model we are going to train, can learn the underlying patterns in the data efficiently and can make accurate predictions on the unseen data points.

5) Why is feature scaling not necessary for the tree-based models ?

Answer: The tree-based models are not sensitive to the scale of the features. If we consider a decision tree algorithm, it splits a node based on a single feature and this is not influenced by the other features, i.e., there won't be any effect of the remaining features if a split is performed based on one single feature.

6) How was the prediction done?

Answer: Based on the performances of each model, I chose the XGBoost regressor model. I considered only the top 6 features as per the feature importance by the model and rebuilt it. So, when a user inputs the age of the concrete specimen. The model takes these values as an input and makes a prediction of the compressive strength of the given concrete specimen.

7) What are the different stages of deployment?

Answer: First, deploy the model locally using Flask (a micro web framework) which works as a backend application. The frontend application is a web page designed using HTML5 with CSS and Bootstrap styling. So, when a user enters the data and hit "predict" button, model in the The backend flask application makes predictions and it will be displayed in the frontend application which the user can take a note of. Then, deployed this application on the web using Azure Cloud

8) How are logs managed?

Answer: We are using different logs as per the steps that we follow in validation and modeling like File validation log, Data Insertion, Model Training log, prediction log etc.