Low Level Design

Air Quality Index

Written By	Milind Sai
Document Version	0.3
Last Revised Date	31-August 2021

Document Control

Change Record:

Version	Date	Author	Comments
0.1	29 – Aug - 2021	Noorudin Shaikh	Introduction & Architecture defined
0.2	30 – Aug - 2021	Karthik Bhargav	Architecture & Architecture Description appended and updated
0.2	30 – Aug 2021	Saurabh Jejurkar	Architecture & Architecture Description appended and updated
0.3	31 – Aug - 2021	Milind Sai	Unit Test Cases defined and appended

Reviews:

Version	Date	Reviewer	Comments

Approval Status:

Version	Review Date	Reviewed By	Approved By	Comments

Contents

1. Introduction

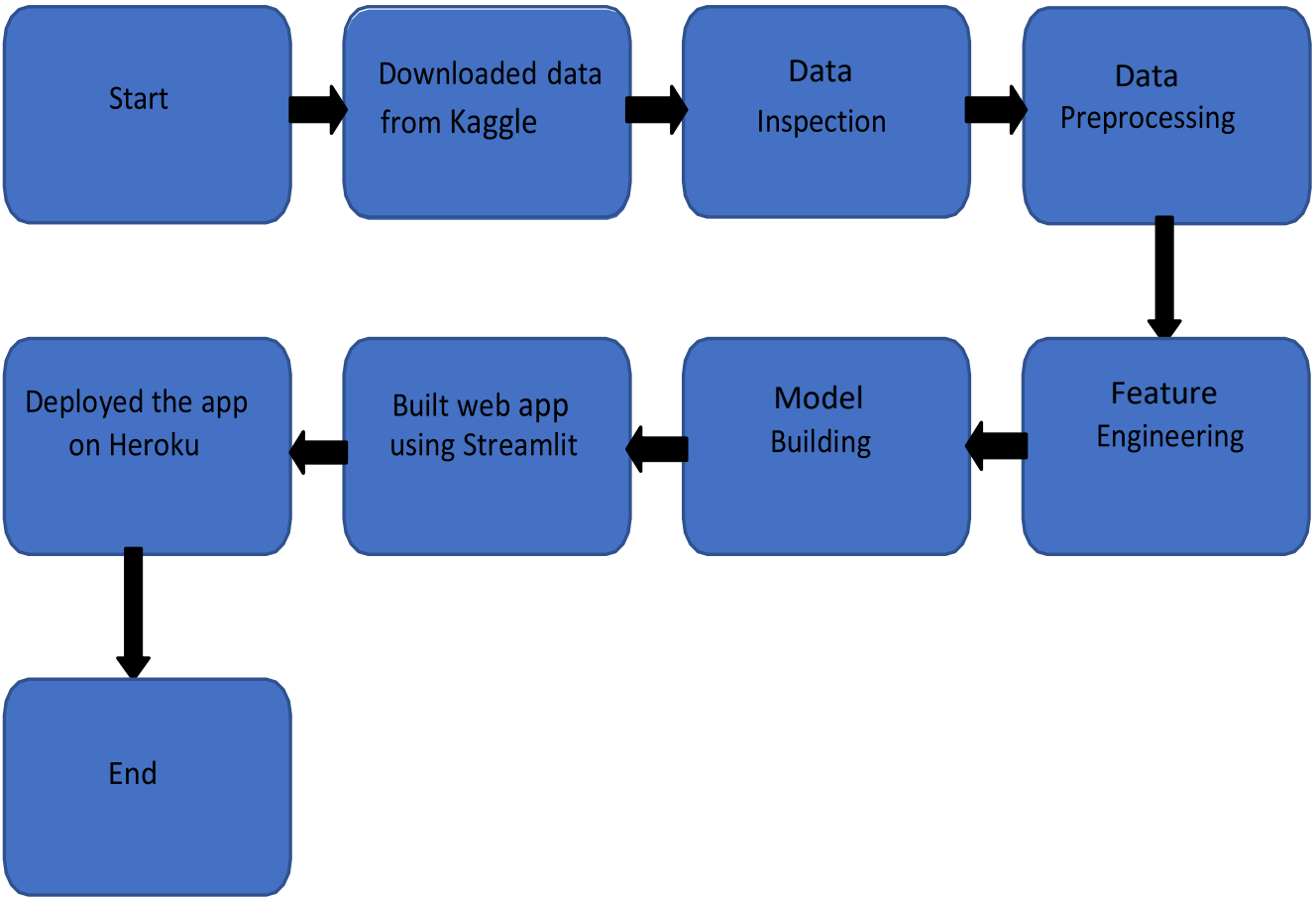
1.1. What is Low-Level design document?

The goal of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual program code for Air Quality Index. 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.

2. Architecture



3. Architecture Description

3.1. Data Description

Context: Air is what keeps humans alive. Monitoring it and understanding its quality is of immense importance to our well-being.
Content: The data set contains air quality data and AQI (Air Quality Index) at daily level (data calculated for a time period of 24-hr) across multiple cities in India.
Data set: Air Quality Data in India(2015-2020) Data file and format: City_day and CSV
Acknowledgements: The data has been made publicly available by the Central Pollution Control Board: https://cpcb.nic.in/ which is the official portal of Government of India. They also have a real-time monitoring app: https://app.cpcbccr.com/AQI_India/

3.2. Data Inspection

In this step, we inspected the data set thoroughly and investigated for NaN values, outliers also checked if any sort of imbalance is present, we plotted histograms for each feature and checked if their distributions were normal or skewed.
Findings: Most of the features which impacted the response variable had right skewed distributions.

3.3. Data Preprocessing

In the preprocessing phase, we dropped the rows containing NaN values, which changed the shape of the data set from 29k to 22k, we can create a data frame (X) consisting of features and a Pandas series(y) consisting of the response variable Air Quality Index.

3.4. Feature Engineering

In this process, we discovered the top 5 features (PM2.5, NO2, CO, SO2 and O3) which had the highest impact on our response variable using the correlation matrix and Extra Tree Regressor model. We then dropped the other features from (X) and only included the top 5 features.

3.5. Model Building

We then split the data into train and test, trained our training data on a variety of ML regression algorithms starting from linear regression, SVR, trees based regression algos. (such as Decision Tree,Random Forest and Xgboost) and going all the way up to building our own ANN models suited for regression. For every model built, we examined the r2 score generated by the model on both training and test data, we also evaluated the performance of each model with evaluation metrics being MAE, MSE and RMSE. Random Forest Regressor after getting hyper-tuned using Random Search CV delivered the ideal results ,i.e , an r2 score of 0.91 on both train and test data. So, we finally settled upon the hyper-tuned Random Forest Regressor as the optimal ML model for our problem.

3.6. Web Application using Streamlit

Streamlit is an open-source python library turns data scripts into shareable web apps. So, for designing ourweb application we will using streamlit.

3.7. Deployment

We will be deploying the model to Heroku.

Web application: airqualityindexchecker

1. 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 URLis accessed	1. Application URLis accessible 2. Application isdeployed	The Application should load completely for the user when theURL is accessed
Verify whether user is able to see inputfields on logging in	Application is accessible	User should be able to see inputfields on logging in
Verify whether user is able to edit allinput fields	Application is accessible	User should be able to edit all inputfields
Verify whether user gets calculate AQI button to submitthe inputs	Application is accessible	User should get calculate AQIbutton to submit the inputs
Verify whether user is presented withthe predicted results on clicking the calculate AQI button.	Application is accessible	User should be presented with results on clicking calculate AQIbutton.
Verify whether the user is able to access the explore page from the sidebar page selection box.	Application is accessible	User should be presented with a sample of a data set followed by plots corresponding to EDA.

Screenshot of the web application