**6. DATA MODELLING**

Predictive modeling in data science is a process of using data and statistical algorithms to make predictions about future outcomes or events. It's a subset of machine learning and is commonly used to forecast trends, behaviors, or events based on patterns and relationships discovered in data. Modeling techniques are based around the use of algorithms. We have to choose an appropriate algorithm or model to use for the prediction.

Data Modelling thus helps to increase consistency in naming, rules, semantics, and security. This, in turn, improves data analytics. The process of data modelling also enforces business rules, regulatory compliances, and government policies on the data. The emphasis is on the need for availability and organisation of data, independent of the manner of its application.

The choice of the model depends on the nature of the data and the problem you're trying to solve. Classification and Regression are two major prediction problems that are usually dealt with in data science.  The key distinction between Classification vs Regression algorithms is Regression algorithms are used to determine continuous values such as price, income, age, etc. and Classification algorithms are used to forecast or classify the distinct values such as True or False, Male or Female, Spam or Not Spam, etc. Common algorithms include linear regression, logistic regression, K-NN, decision trees, random forests and support vector machines.

In our dataset, we have two target variables, AQI and AQI\_Bucket. So we established both regression and classification algorithms. For the regression problem, the feature ‘AQI’ is the target variable, and for the classification, ‘AQI\_Bucket’ is the target variable. As the initial step, the data has been split into training data and testing data, with a test size of 0.2. The subsequent task is to determine the most suitable algorithm, which can be challenging. In order to identify the most appropriate algorithm, the accuracy scores for different algorithms have been evaluated.

We treated outliers using different methods. So the model trained by each one is different and checked for accuracy. Model creation is done before and after the scaling of the dataset. Finally, we decided to go with the outlier treatment of flooring and capping with minimum (upper limit) and maximum (lower limit) values by the method of interquartile range. Different regression models were created and checked for accuracy.

|  |  |
| --- | --- |
| **Regression Algorithm** | **R2\_score** |
| Linear Regression | 0.5881 |
| Lasso Regression | 0.5881 |
| Support vector regressor | 0.5502 |
| Decision tree regressor | 0.8320 |
| Random forest regressor | 0.9133 |

**Figure 28 - Comparison of accuracy score for various models**

The table illustrates a comparison of accuracy scores for different models. The selection of the model is based on the evaluation metrics and cross-validation results, aiming to identify the model with the highest accuracy. In this case, Figure 28 indicates that the Random Forest Regressor achieves the highest accuracy of 0.9133, making it the best model for the given task. Hence, Random Forest Regressor is selected for deploying the model.

Random Forest is an ensemble learning method that combines multiple decision trees. Each tree is trained on a random subset of features and the final prediction is made by aggregating the predictions of individual trees. It's known for its robustness and high accuracy.

**6.1 Hyperparameter Tuning**

A Machine Learning model is defined as a mathematical model with a number of parameters that need to be learned from the data. By training a model with existing data, we are able to fit the model parameters.

Hyperparameter tuning is basically referred to as tweaking the parameters of the model, which is basically a prolonged process. Some set of parameters that are used to control the behaviour of the model/algorithm and adjustable in order to obtain an improvised model with optimal performance is so-called Hyperparameters.

In the case of random forest there are parameters which either to increase the predictive power of the model or to make it easier to train the model. Following are the parameters we will be talking about in more detail.

We will try adjusting the following set of hyperparameters:

* n\_estimators = number of trees in the forest
* max\_features = max number of features considered for splitting a node
* max\_depth = max number of levels in each decision tree
* min\_samples\_split = min number of data points placed in a node before the node is split
* min\_samples\_leaf = min number of data points allowed in a leaf node

After the hyperparameter tuning of the random forest regressor, we observed that the accuracy of the model was reduced. So we decided to go with the model without tuning, which already has an accuracy of 0.9113.

1. **MODEL DEPLOYMENT**

Model deployment is simply the engineering task of exposing an ML model to real use. Deployment is the method by which you integrate a [machine learning](https://www.datarobot.com/wiki/machine-learning/) model into an existing production environment to make practical business decisions based on data. It is one of the last stages in the [machine learning life cycle](https://www.datarobot.com/wiki/machine-learning-life-cycle/) and can be one of the most cumbersome.

Model deployment is one of the most difficult processes of gaining value from machine learning. It requires coordination between data scientists, IT teams, software developers, and business professionals to ensure the model works reliably in the organization’s production environment. This presents a major challenge because there is often a discrepancy between the programming language in which a machine learning model is written and the languages your production system can understand, and re-coding the model can extend the project timeline by weeks or months.

In order to get the most value out of machine learning models, it is important to seamlessly deploy them into production so a business can start using them to make practical decisions.

**7.1 Flask**

Flask is a web application framework written in python, in simple terms it helps end users interact with your python code (in this case our ML models) directly from their web browser without needing any libraries, code files, etc. Flask enables you to create web applications very easily.

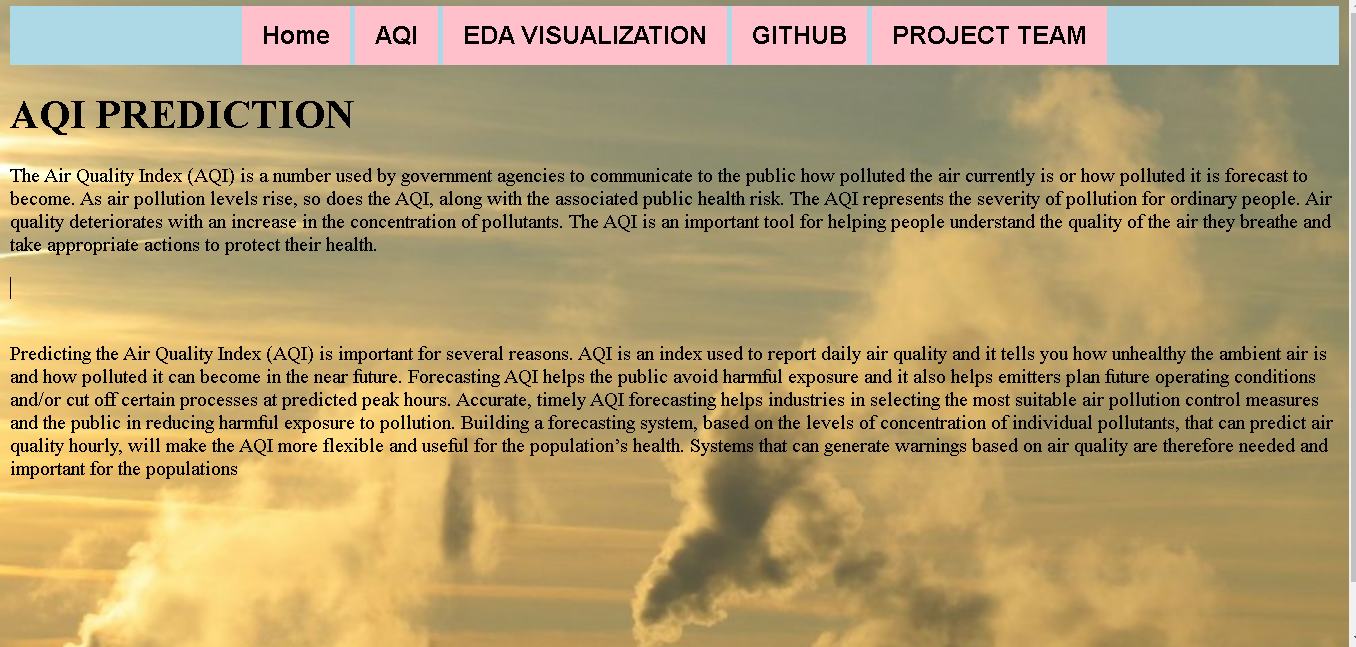
**7.2 Directory Structure**

Directory structure will give us a broader picture of the overall project and it is also useful to know it when you working with flask. The project is saved under a main directory.

* templates: This folder contains the html files (index.html, predict.html) that would be used by our main file (app.py) to generate the front end of our application.
* static: This folder contains two folders named CSS and images. CSS files are saved in CSS folder.
* app.py: This is the main application file, where all our code resides and it binds everything together.
* model.pkl: This is our regression model that we would be using, in this case it is a Random Forest Regression Model.

**7.3 Front End Development**

The front end is created by using HTML. HTML is a markup language that defines the structure of your content. HTML consists of a series of [elements](https://developer.mozilla.org/en-US/docs/Glossary/Element), which you use to enclose, or wrap, different parts of the content to make it appear a certain way, or act a certain way.

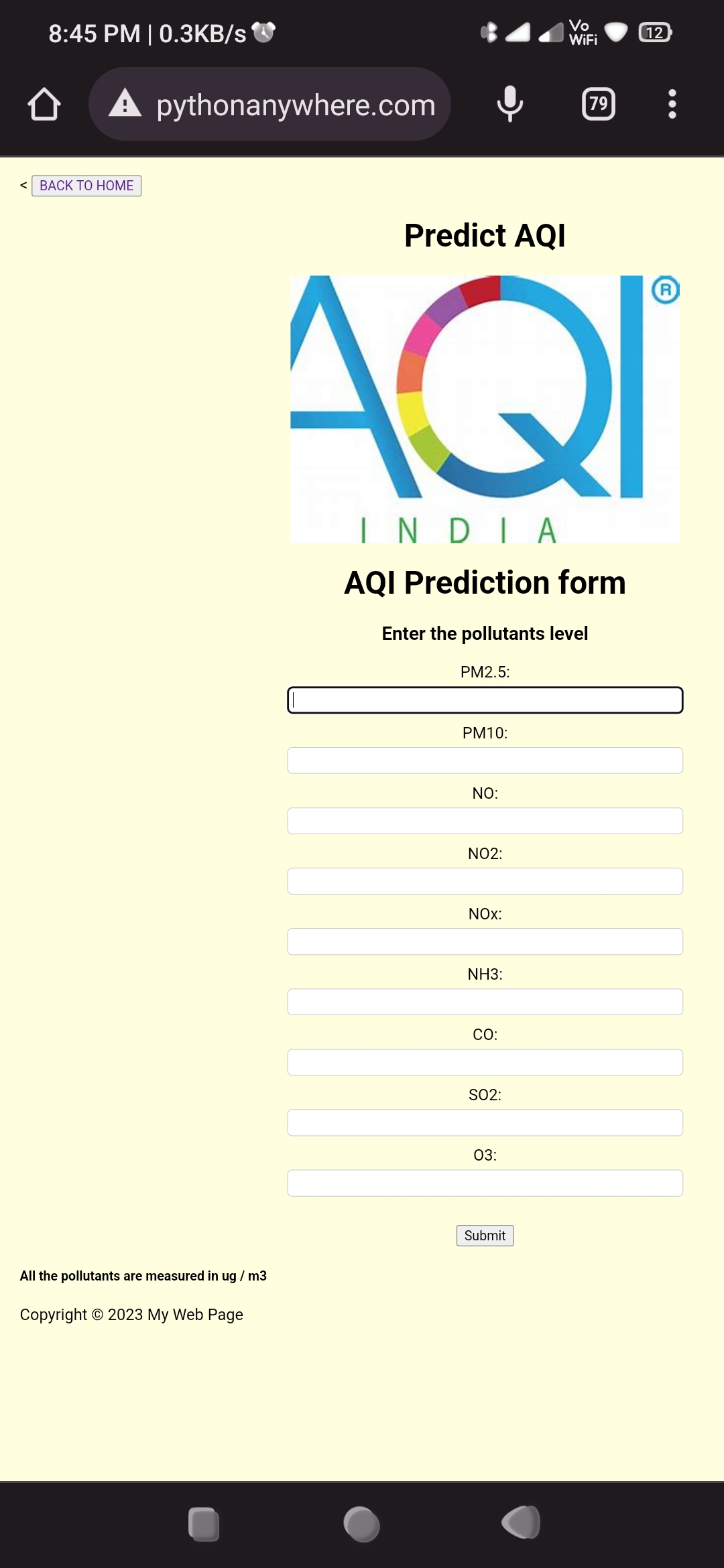


**Figure 29 – Front end**

**7.4 Web Deployment**

The web application developed using Flask allows users to predict Air Quality Index based on Random Forest Regressor. By providing relevant input parameters, users can obtain predictions about which category of air pollution the area belongs to.

User input: The web application prompts users to input specific parameters required for prediction. These parameters may include PM2.5, PM10, NO, NO2, SO2, O3, NOx, CO & NH3 pollutants measurements. By collecting this information the model can make accurate predictions based on the trained Random forest Algorithm.



**Figure 30 – AQI prediction form**

Model-Prediction: Once the user provides the necessary input, the web application passes this data to the Random Forest model for prediction. The model utilizes its learned patterns and relationships to generate a prediction regarding the AQI value and subsequently its category. The prediction is then displayed to the users through the web interface.

1. **RESULT**

The selection of the model is based on the evaluation metrics and cross-validation results, aiming to identify the model with the highest accuracy. In this case **Random Forest Regressor** achieves the highest accuracy of 0.9133, making it the best model for the given task. Hence, Random Forest Regressor is selected for deploying the model. Then web hosting was done to predict the AQI values and their category. The webpage was created using HTML and the page was hosted using **pythonanywhere**.

## Web Hosting - URL

<http://dsaproject.pythonanywhere.com/>

**Source Code-GitHub link**

[aliyapsubeer/DSA\_FINAL\_PROJECT (github.com)](https://github.com/aliyapsubeer/DSA_FINAL_PROJECT)

**Challenges Faced:**

* Data preprocessing: Preprocessing the AQI dataset is an important step in the analysis process. [This involves selecting key features through correlation analysis, developing insights into various hidden patterns in the dataset, and identifying pollutants that directly affect the AQI](https://link.springer.com/article/10.1007/s13762-022-04241-5).
* [Data imbalance: AQI prediction datasets may suffer from data imbalance problems, which can be solved using resampling techniques](https://link.springer.com/article/10.1007/s13762-022-04241-5).
* Feature selection: Selecting the right features to use in the prediction model is a critical step in the analysis process.
* Model selection: Choosing the right prediction model is another important step in the analysis process. [Several machine learning models can be employed to predict air quality, and their performances can be evaluated and compared using established performance parameters](https://link.springer.com/article/10.1007/s13762-022-04241-5).
* Feature engineering: Feature engineering is a critical step in AQI prediction, as it helps to capture the potential features from the AQI series. However, this can be a challenging task, as it involves an “extract-merge-filter” procedure to generate candidate feature sets.
* Model matching: In order to achieve accurate AQI prediction, it is important to match predictors with their optimal features1. This can be a challenging task, as it requires the use of comprehensive metrics to evaluate the performance of different predictors on different feature sets.