BRAC
UNIVERSITY

Inspiring Excellence

CSE 422: ARTIFICIAL INTELLIGENCE

Fall 2024

Project Report

AIR QUALITY PREDICTOR

Student Information:

Name : B M Rauf

Student ID : 22201782

Department of Computer Science and Engineering

Brac University

Table of Contents

Introduction

Air quality prediction is a critical task in environmental science and public health management. This project aims to classify air quality into different categories based on various environmental factors using machine learning models. The dataset contains attributes such as temperature, humidity, pollutant levels (PM2.5, PM10, NO2, SO2, CO), proximity to industrial areas, and population density, which serve as predictors for the classification task.

The goal is to preprocess the data, scale the features, split the dataset, and train multiple models to identify the most effective one for the classification task. With increasing urbanization and industrial activities, air quality monitoring is essential to ensure public safety and mitigate health risks associated with poor air quality.

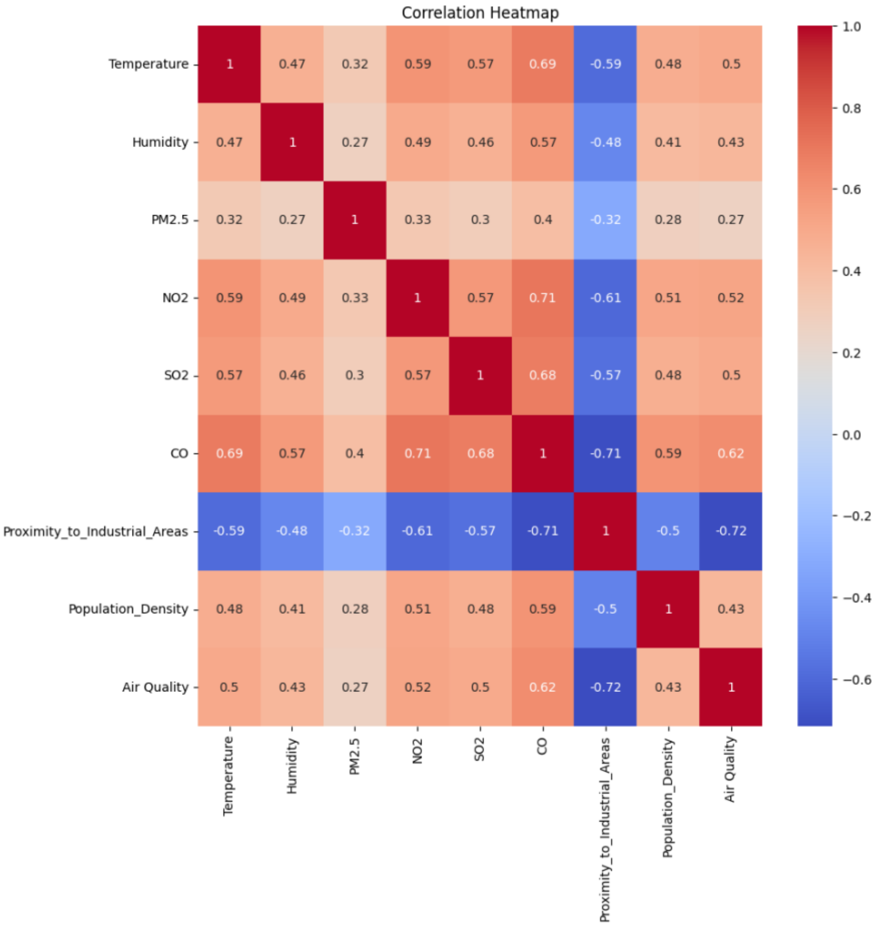
Dataset Description

- **Source:**

  The dataset was sourced from Kaggle, containing information related to air pollution and its contributing factors.

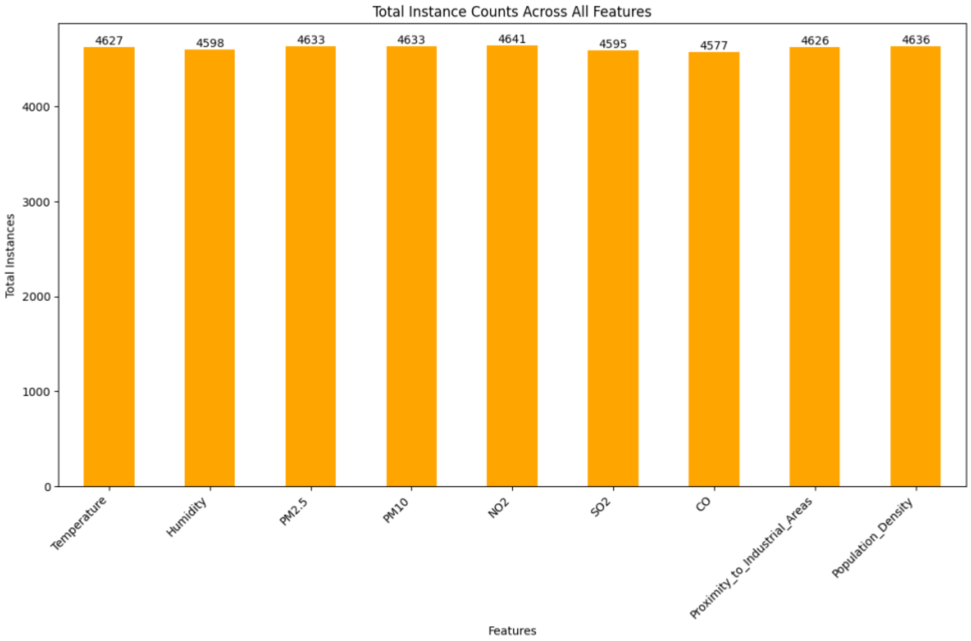
  - Link: Air Quality Dataset
- **Dataset Description:**
  - **Number of Features:** 9 features, including temperature, humidity, pollutant levels, and population density.
  - **Problem Type:** Classification problem, as the target is categorical (Good, Moderate, Poor, Hazardous).
  - **Number of Data Points:** 5,000 samples.

- **Feature Types:** All features are numerical except for the target variable, which was encoded into numerical values using LabelEncoder.
- **Correlation Analysis:** A heatmap was generated to visualize correlations, highlighting highly correlated features that might impact the model.



- **Unbalanced Dataset:**

The dataset is not perfectly balanced because the number of instances varies across features. Here, the instance counts range from **4577 to 4641**, meaning some features have slightly more or fewer values than others.



Dataset Preprocessing

- **Faults:**
  - **NULL Values:** The dataset did not contain any null values initially. However, we randomly inserted 5–10% null values for testing purposes.
  - **Categorical Values:** The target variable ("Air Quality") was categorical.
- **Solutions:**
  - **Dropped Columns:** The **'PM10'** column was removed due to its high correlation with **'PM2.5'** to reduce redundancy. Missing values in categorical data, specifically the **'Air Quality'** column, were filled using the **mode** (most frequent value), while missing values in numerical columns were replaced with the **median** (middle value) to preserve data distribution.

Finally, any remaining rows with null values after this process were dropped to ensure data completeness.

- **Encoding:** The target variable ("Air Quality") was encoded using LabelEncoder to convert categorical values into numeric representations.

Feature Scaling

Feature scaling is essential for ensuring that machine learning algorithms treat all features equally. The dataset was scaled using the MinMaxScaler from scikit-learn, which normalizes the values between 0 and 1. This step ensures the dataset is suitable for algorithms sensitive to magnitude differences in features.

Dataset Splitting

The dataset was split into training and testing subsets to evaluate model performance:

- Splitting Technique: Stratified Random Sampling was used to ensure that the distribution of the target variable in the training and testing subsets matches the original dataset.
- Training Set: 70% of the data.
- Testing Set: 30% of the data.

The splitting ensured an unbiased evaluation of the model's performance on unseen data while maintaining the representativeness of the target variable distribution. The random_state parameter was used to ensure reproducibility of the splits.

Model Training & Testing

Multiple machine learning models were trained to classify air quality, including:

1. **Logistic Regression**
   - **Description:**

     Logistic Regression is a statistical model used for binary and multiclass classification tasks. It estimates probabilities using a logistic function, which maps the input features to a probability value between 0 and 1. The output is then categorized into classes based on a threshold (commonly 0.5).

   - **How It Works:**
     - **Logistic Function:** The algorithm uses the logistic (sigmoid) function:

       $$P(y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_n X_n)}}$$

       Where $X_1, X_2, \ldots, X_n$ are the input features, and $\beta_0, \beta_1, \ldots, \beta_n$ are the coefficients that the model learns during training.

     - **Training:** During training, the model optimizes the coefficients (β) to minimize the loss function, typically the log-loss (cross-entropy loss), which measures how well the predicted probabilities match the actual class labels.
     - **Prediction:** For a new data point, the model calculates the probability of belonging to each class. If the probability

exceeds the threshold (e.g., 0.5), the data point is classified as one class; otherwise, it is classified as another.

  - **Multiclass Classification:** For multiclass problems, Logistic Regression can use techniques like One-vs-Rest (OvR) or SoftMax regression to handle multiple classes.

2. **Naive Bayes**

- **Description:**

  Naive Bayes is a probabilistic classifier based on Bayes' theorem , assuming independence between features. Despite its simplicity and "naive" assumption of feature independence, it performs surprisingly well in many real-world scenarios, especially in text classification and spam filtering.

- **How It Works:**

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}$$

  - **Bayes' Theorem**:

    Where:

    - P(y|X): Posterior probability of class y given the features X.
    - P(X|y): Likelihood of observing the features X given the class y.
    - P(y): Prior probability of class y.

        - P(X): Probability of observing the features X (normalizing constant).
    - **Training:**
        - The model calculates the prior probabilities P(Y) for each class and the likelihoods P(X|Y) for each feature given the class.
        - Since Naive Bayes assumes feature independence, the joint probability of all features given the class is calculated as the product of individual feature probabilities:
            - $P(X|Y)=P(X_1|Y)\cdot P(X_2|Y)\cdot\ldots\cdot P(X_N|Y)$
    - **Prediction:**
        - For a new data point, the model calculates the posterior probability P(Y|X) for each class using Bayes' theorem.
        - The class with the highest posterior probability is chosen as the predicted class.

3. **Random Forest Classifier**
    - **Description:**

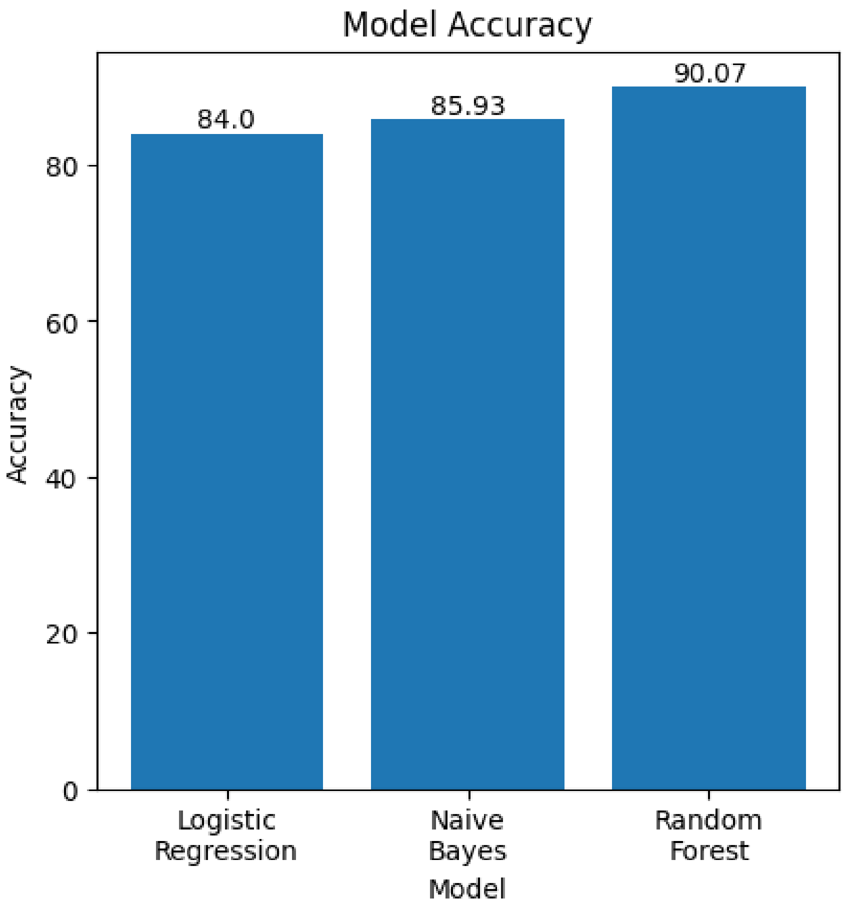
      Random Forest is an ensemble learning method that builds multiple decision trees during training and aggregates their predictions (e.g., majority voting for classification) to improve accuracy and reduce overfitting. It is robust to noise and works well on large datasets.

    - **How It Works:**

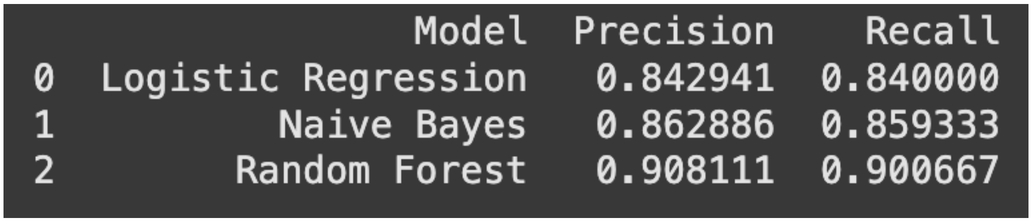
- **Bootstrap Sampling:** Random Forest creates multiple decision trees by sampling the dataset with replacement (bootstrap sampling). Each tree is trained on a different subset of the data.
- **Random Feature Selection:** At each split in a tree, only a random subset of features is considered. This introduces diversity among the trees and reduces the risk of overfitting.
- **Tree Construction:** Each tree is grown fully without pruning, ensuring maximum depth and complexity.
- **Aggregation:** For classification tasks, the final prediction is made by majority voting among all the trees. Each tree "votes" for a class, and the class with the most votes is selected as the final prediction.
- **Advantages:**
  - Reduces overfitting compared to individual decision trees.
  - Handles high-dimensional data well.
  - Provides feature importance scores, helping identify the most influential features.

Model Selection/Comparison Analysis

- Bar Chart: Showcased the prediction accuracy of all models.

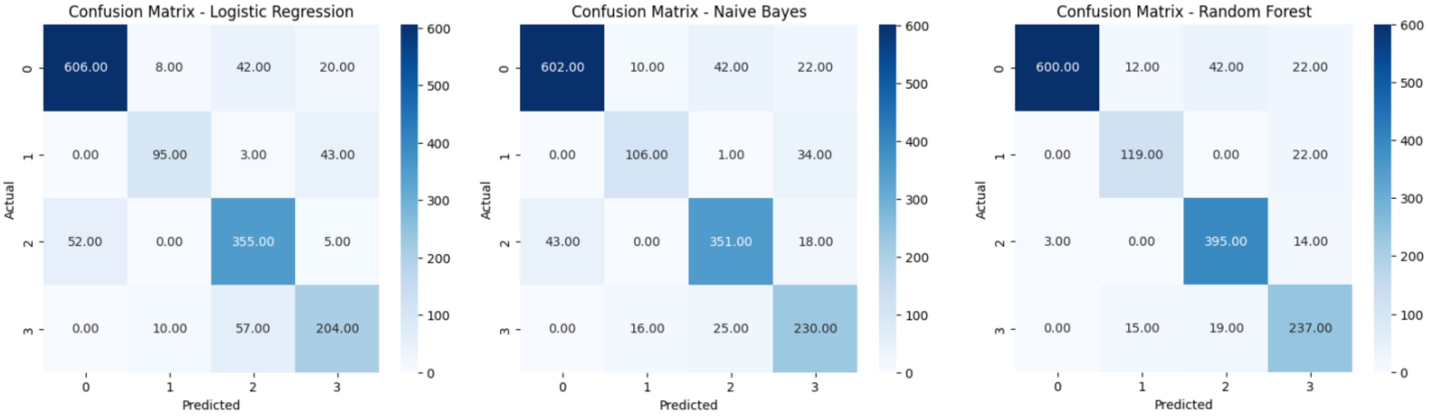


- Precision and Recall:
  - Precision: The proportion of correctly predicted positive instances out of all predicted positives. It measures accuracy in positive predictions.
  - Recall: The proportion of correctly predicted positive instances out of all actual positives. It measures the ability to find all positive instances.



```
                 Model  Precision    Recall
0  Logistic Regression   0.842941  0.840000
1          Naive Bayes   0.862886  0.859333
2        Random Forest   0.908111  0.900667
```

- Confusion Matrix: Analyzed for each model to evaluate classification performance in detail.



Conclusion

The Air Quality Predictor project successfully used machine learning to classify air quality into categories such as Good, Moderate, Poor, and Hazardous. Through proper preprocessing, scaling, and splitting of the dataset, the models — Logistic Regression, Naive Bayes, and Random Forest — achieved accuracies of 84.0%, 85.93%, and 90.07%, respectively. Metrics like precision, recall, and confusion matrix confirmed their effectiveness. This project demonstrates how machine learning can be a reliable tool to monitor air quality and prevent health risks associated with pollution.