AIR QUALITY: COMPARATIVE ANALYSIS OF MODELS

Title of Project: Air Quality: Comparative Analysis of Models

Student Name: HARSH KUMAR JHA

Enrolment Number: 02219011921

Email ID: rahkj1000@gmail.com

Contact Number: +91 98914 42744

Google Colab Link: https://drive.google.com/file/d/1F_YyyqT-F_PL07q_OayjtWJ6TudJcFBm/view?usp=sharing OR (https://tinyurl.com/air-quality-harsh-kumar-jha)

Github Link: https://github.com/hkjhacode/Air-Quality-Control-Regression-Analysis-Using-Different-Models OR https://tinyurl.com/air-quality-harsh-jha-github

AIR QUALITY: COMPARATIVE ANALYSIS OF MODELS

ABSTRACT:

This report presents a comprehensive analysis of air quality data collected from an array of chemical sensors deployed in a polluted area within an Italian city. The dataset provides hourly averaged sensor responses and corresponding ground truth concentrations for various air pollutants such as CO, Non Metanic Hydrocarbons, Benzene, Total Nitrogen Oxides (NOx), and Nitrogen Dioxide (NO2). The dataset spans a one-year period from March 2004 to February 2005.

The main objective of this analysis is to develop a regression model that can accurately estimate the concentration of CO in the air based on the sensor responses. The dataset contains missing values tagged with -200, which are replaced with NaN values for further processing. Rows with missing target values (CO(GT)) are dropped from the dataset.

To investigate the missing data, a bar chart is plotted to visualize the count and percentage of missing values for each attribute. This helps in identifying attributes with significant amounts of missing data.

Outlier detection is performed using box plots for each attribute to identify any extreme values that might affect the model's performance.

To handle missing values, a SimpleImputer is used to replace NaN values with the mean of each attribute.

The dataset is then split into training and testing sets for model development and evaluation. Numeric attributes are selected for scaling using a RobustScaler, which ensures robustness to outliers.

Various regression models are trained and evaluated, including Linear Regression, Decision Tree Regression, Random Forest Regression, Support Vector Regression, Lasso Regression, Ridge Regression, Elastic Net, K-Nearest Neighbors Regression, XGBoost Regression, LightGBM Regression, and CatBoost Regression. Evaluation metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R2 score are calculated to assess the performance of each model.

The results of the analysis provide insights into the performance of different regression models in estimating the concentration of CO in the air. The models' strengths and weaknesses are highlighted, aiding researchers and practitioners in selecting the most suitable model for similar air quality estimation tasks.

In conclusion, this analysis demonstrates the application of machine learning models to estimate air quality based on sensor responses. The findings contribute to the understanding of the performance of different regression models in this domain and can inform the development of air quality monitoring systems and pollution mitigation strategies.

Keywords:

1. Air quality dataset
2. Metal oxide chemical sensors
3. Air Quality Chemical Multisensory Device
4. Polluted area
5. Road-level air pollution
6. Italian city air quality
7. Ground Truth concentrations
8. Concept drift and sensor drift

Introduction:

Air quality monitoring is crucial for assessing and mitigating the impact of pollution on public health and the environment. In this report, we delve into the analysis of air quality data using the Air Quality Chemical Multisensor Device dataset. This dataset provides hourly averaged responses from metal oxide chemical sensors, along with reference analyzer measurements, allowing us to estimate pollutant concentrations accurately.

The dataset covers a one-year period from March 2004 to February 2005, providing a comprehensive record of air quality in a significantly polluted area within an Italian city. With 9358 instances, it represents one of the longest freely available recordings of on-field deployed air quality chemical sensor devices. The recorded pollutants include carbon monoxide (CO), non-methanic hydrocarbons (NMHC), benzene, total nitrogen oxides (NOx), and nitrogen dioxide (NO2).

Before diving into the analysis, we perform data preprocessing steps to ensure data quality and handle missing values. We concatenate the date and time columns to create a unified datetime attribute, enabling temporal analysis. We replace missing values, tagged with -200, with NaN for ease of handling. Rows with missing target values for CO are dropped from the dataset.

To gain insights into the missing data, we visualize the count and percentage of missing values for each attribute. This helps us identify the attributes with significant data gaps, which may require additional handling techniques.

Next, we conduct outlier detection using box plots for each attribute. Outliers can distort the statistical analysis and model training process, so identifying and addressing them is essential for accurate predictions. The box plots allow us to visualize the distribution of data and identify any extreme values that might indicate outliers.

After handling missing values and outliers, we proceed with feature scaling to ensure that all attributes have similar units. We employ the RobustScaler to scale the numeric columns in the dataset. This transformation helps prevent attributes with large value ranges from dominating the analysis and model training process.

With the preprocessed data, we move on to the core of the analysis – training and evaluating regression models to predict CO concentrations. We consider a range of regression algorithms, including linear regression, decision tree regression, random forest regression, support vector regression, Lasso regression, Ridge regression, ElasticNet regression, K-nearest neighbors regression, XGBoost regression, LightGBM regression, and CatBoost regression. For each model, we evaluate its performance using metrics such as mean squared error (MSE), mean absolute error (MAE), root mean squared error (RMSE), and R2 score.

By comparing the performance of various regression models, we aim to identify the most suitable approach for predicting CO concentrations based on the provided sensor responses and auxiliary data. The selected model can serve as a valuable tool for air quality monitoring and management efforts.

In conclusion, this report showcases an in-depth analysis of the Air Quality Chemical Multisensor Device dataset, focusing on predicting CO concentrations. Through data preprocessing, outlier detection, and regression model training, we aim to contribute to the understanding and monitoring of air pollution, ultimately aiding in the development of effective strategies for air quality improvement and public health protection.

Dataset: Air Quality:

About the Dataset:

The Air Quality dataset contains hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an Air Quality Chemical Multisensor Device. The device was placed in a significantly polluted area at road level within an Italian city. The data was recorded from March 2004 to February 2005, representing a one-year period.

The dataset provides measurements of various air pollutants, including CO (Carbon Monoxide), Non Methanic Hydrocarbons, Benzene, Total Nitrogen Oxides (NOx), and Nitrogen Dioxide (NO2). The ground truth hourly averaged concentrations for these pollutants were obtained from a co-located reference certified analyzer.

The dataset consists of 9358 instances, with each instance containing the following attributes:

1. Date: The date in DD/MM/YYYY format.
2. Time: The time in HH.MM.SS format.
3. True hourly averaged concentration CO: The reference analyzer's measured concentration of CO in mg/m^3.
4. PT08.S1 (tin oxide): Hourly averaged sensor response targeted at CO.
5. True hourly averaged overall Non Metanic HydroCarbons concentration: The reference analyzer's measured concentration of Non Metanic Hydrocarbons in microg/m^3.
6. True hourly averaged Benzene concentration: The reference analyzer's measured concentration of Benzene in microg/m^3.
7. PT08.S2 (titania): Hourly averaged sensor response targeted at NMHC.
8. True hourly averaged NOx concentration: The reference analyzer's measured concentration of NOx in ppb.
9. PT08.S3 (tungsten oxide): Hourly averaged sensor response targeted at NOx.
10. True hourly averaged NO2 concentration: The reference analyzer's measured concentration of NO2 in microg/m^3.
11. PT08.S4 (tungsten oxide): Hourly averaged sensor response targeted at NO2.
12. PT08.S5 (indium oxide): Hourly averaged sensor response targeted at O3.
13. Temperature: Temperature in °C.
14. Relative Humidity: Relative humidity in percentage.
15. AH: Absolute Humidity.

The dataset contains missing values, which are tagged with -200. These missing values are replaced with NaN (Not a Number) to handle them effectively.

Data used in this notebook can be found and downloaded from:
https://archive.ics.uci.edu/dataset/360/air+quality

PREPROCESSING:

The provided code performs several data preprocessing steps on the air quality dataset before applying machine learning models. Here is a summary of the preprocessing steps:

1. Loading the Dataset:

The dataset is loaded from the provided CSV file using pandas.

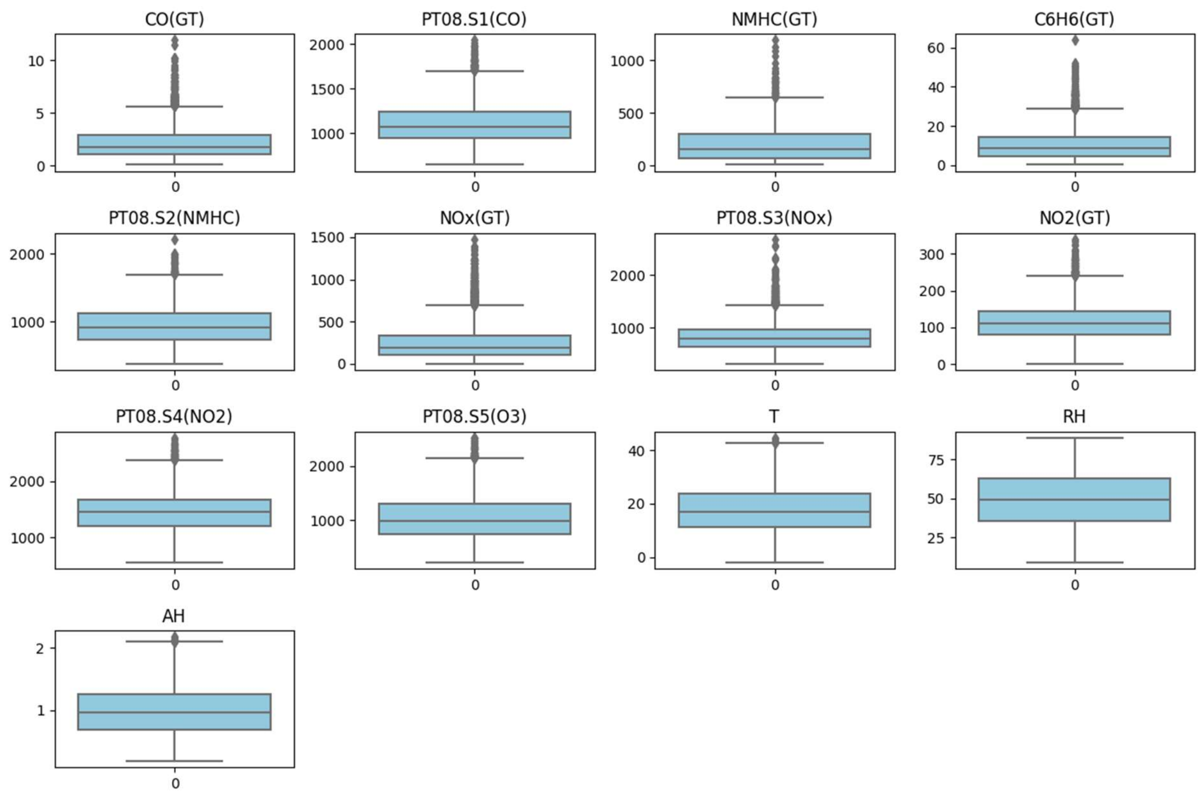
2. Concatenating Date and Time:

The 'Date' and 'Time' columns are combined to create a new attribute called 'Datetime'. The values are converted to the datetime format for easier manipulation.

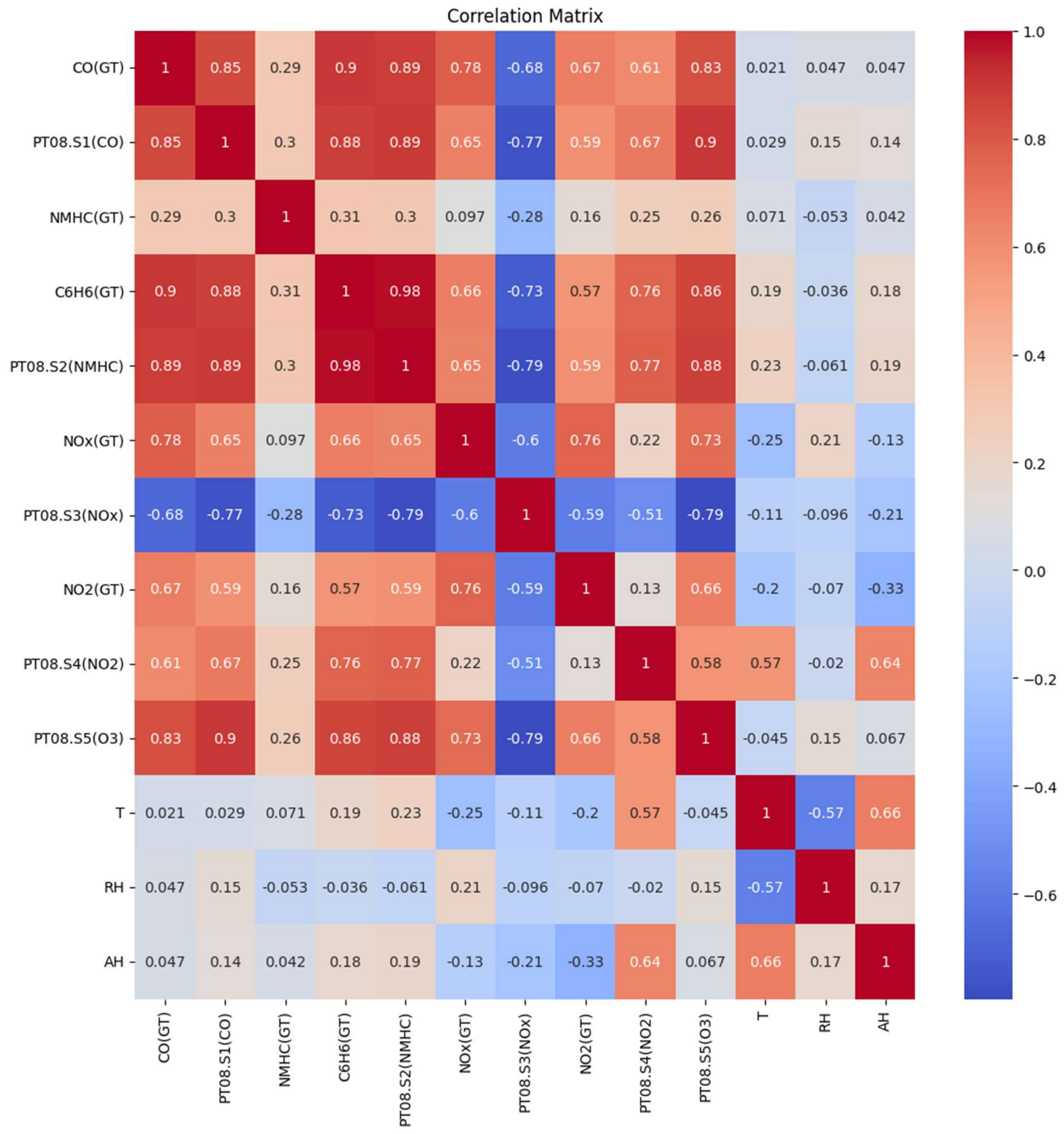
3. Handling Missing Values:

Missing values in the dataset are tagged with a value of -200. These missing values are replaced with NaN (Not a Number) using the 'replace' function.

Box plot for outlier detection in attributes:



Correlational matrix:



4. Dropping Rows with Missing Target Values:

Rows with missing values in the target variable 'CO(GT)' are dropped from the dataset using the 'dropna' function.

5. Analyzing Missing Values:

The code selects specific attributes that contain NaN values and calculates the count and percentage of missing values for each attribute. The results are stored in a dataframe and visualized using bar charts.

6. Handling Missing Values (Imputation):

The missing values in the dataset are imputed using the SimpleImputer from scikit-learn. The 'mean' strategy is used to replace the missing values with the mean of each column.

7. Scaling Numeric Attributes:

The numeric columns in the dataset are selected, and the RobustScaler from scikit-learn is used to scale the numeric attributes. This step helps to standardize the units and make the attributes comparable.

8. Defining Regressor Models:
Several regression models are imported from scikit-learn and other libraries, including Linear Regression, Decision Tree, Random Forest, Support Vector Machine, Lasso, Ridge, ElasticNet, K-Nearest Neighbors, XGBoost, LightGBM, and CatBoost.

9. Training and Evaluating Models:

The code loops through each regressor model, fits it on the training data, makes predictions on the test data, and calculates various evaluation metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R2 Score. The results are stored in a dictionary.

10. Printing Model Metrics:

Finally, the code prints the evaluation metrics for each model, providing insights into their performance.

The preprocessing steps mentioned above address missing values, handle attribute scaling, and prepare the data for training machine learning models.

After solving these issues dataset is ready for machine learning model (dataset does have any missing value having all attributes are in float (no objects). After this next step will be to choose target values.

FEATURES MODELING:

In this project, we are working with a dataset that aims to predict the air quality based on various sensor measurements. The target variable for our modeling task is the concentration of Carbon Monoxide (CO) in mg/m^3, represented by the attribute 'CO(GT)'.

We begin the modeling process by selecting the dependent attributes, which are the input features used to predict the target variable. In this case, we have a set of independent attributes that include sensor responses, temperature, relative humidity, and absolute humidity. The independent attributes chosen for modeling are as follows:

1. PT08.S1(CO): Hourly averaged sensor response of tin oxide (CO-targeted)

2. NMHC(GT): True hourly averaged overall Non Metanic HydroCarbons concentration inmicrog/m^3

3. C6H6(GT): True hourly averaged Benzene concentration in microg/m^3

4. PT08.S2(NMHC): Hourly averaged sensor response of titania (NMHC-targeted)

5. NOx(GT): True hourly averaged NOx (Total Nitrogen Oxides) concentration in ppb

6. PT08.S3(NOx): Hourly averaged sensor response of tungsten oxide (NOx-targeted)

7. NO2(GT): True hourly averaged Nitrogen Dioxide concentration in microg/m^3

8. PT08.S4(NO2): Hourly averaged sensor response of tungsten oxide (NO2-targeted)

9. PT08.S5(O3): Hourly averaged sensor response of indium oxide (O3-targeted)

10. T: Temperature in °C

11. RH: Relative Humidity (%)

12. AH: Absolute Humidity

These attributes provide information about the different gases and environmental conditions that can impact air quality.

By using these independent attributes, we aim to build a regression model that can accurately predict the concentration of CO.

SPLITING DATA INTO TRAIN AND TEST:

Now, we have input data and target data. Next will be to fit model but Model should be train and test on different data to avoid memorization. So, dividing the whole data into training and testing data (75% data for training and 25% for test). We will fit model on training data and test the model accuracy on test data.

FEATURES SCALING:

Standardization can become skewed or biased if the input variable contains outlier values. To overcome this, the median and interquartile range can be used when standardizing numerical input variables.
The attributes (timestamp, heart rate) are larger value than other values this can lead to biased on training model. To avoid this we Scaled the data by robustScaler.

Why to use RobustScaler ?

The dataset attributes have different units and contains outliers. And robustScaler is not affected by outliers.

The formula of RobustScaler is

$$X_{new} = \frac{X - X_{median}}{IQR}$$

Since, it uses the interquartile range, it absorbs the effects of outliers while scaling. The interquartile range (Q3 — Q1) has half the data point. If you have outliers that might affect your results or statistics and don’t want to remove them, RobustScaler is the best choice.

MODELING:

In this project, we worked with the air quality dataset to develop a regression model to predict the concentration of Carbon Monoxide (CO) in the air. The dataset contains hourly averaged responses from a set of metal oxide chemical sensors, along with reference analyzer measurements for CO and other air pollutants.

We started by loading the dataset using pandas and preprocessing the data. We concatenated the 'Date' and 'Time' columns to create a new attribute 'Datetime' representing the timestamp of each data point. We then converted the 'Datetime' column to the appropriate datetime format. Missing values in the dataset, tagged with -200, were replaced with NaN values. We dropped rows with missing target values (CO concentrations) from the dataset.

Next, we analyzed the missing data in the remaining attributes. We calculated the count and percentage of missing values for each attribute and visualized them using bar plots. This helped us understand the extent of missing data in the dataset.

To handle the missing values, we used the SimpleImputer from scikit-learn to replace the NaN values with the mean of each column. This imputation strategy allowed us to fill in the missing values and ensure that the dataset was ready for modeling.

After handling the missing values, we performed feature scaling using the RobustScaler from scikit-learn. We selected the numeric columns in the dataset and applied the scaling transformation to them. This step was important to ensure that all the attributes were on a similar scale, which can improve the performance of some regression models.

We then proceeded to train and evaluate various regression models. We considered a range of popular regression algorithms, including Linear Regression, Decision Tree Regression, Random Forest Regression, Support Vector Regression, Lasso Regression, Ridge Regression, Elastic Net Regression, K-Nearest Neighbors Regression, XGBoost Regression, LightGBM Regression, and CatBoost Regression.

For each model, we fitted it to the training data and made predictions on the test data. We evaluated the performance of each model using metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R2 Score. These metrics allowed us to assess how well the models predicted the CO concentrations.

RESULT AND CONCLUSION:

After evaluating the regression models on the air quality dataset, we obtained the following results:

- LinearRegression:
  - MSE: 1.36
  - MAE: 0.89
  - RMSE: 1.16
  - R2SCORE: 0.72

- DecisionTreeRegressor:
  - MSE: 1.74
  - MAE: 1.05
  - RMSE: 1.32
  - R2SCORE: 0.63

- RandomForestRegressor:
  - MSE: 1.19
  - MAE: 0.83
  - RMSE: 1.09
  - R2SCORE: 0.75

- SVR:
  - MSE: 2.78
  - MAE: 1.29
  - RMSE: 1.67
  - R2SCORE: 0.35

- Lasso:
  - MSE: 1.36
  - MAE: 0.89
  - RMSE: 1.16
  - R2SCORE: 0.72

- Ridge:
  - MSE: 1.36
  - MAE: 0.89
  - RMSE: 1.16
  - R2SCORE: 0.72

- ElasticNet:
  - MSE: 1.36
  - MAE: 0.89
  - RMSE: 1.16
  - R2SCORE: 0.72

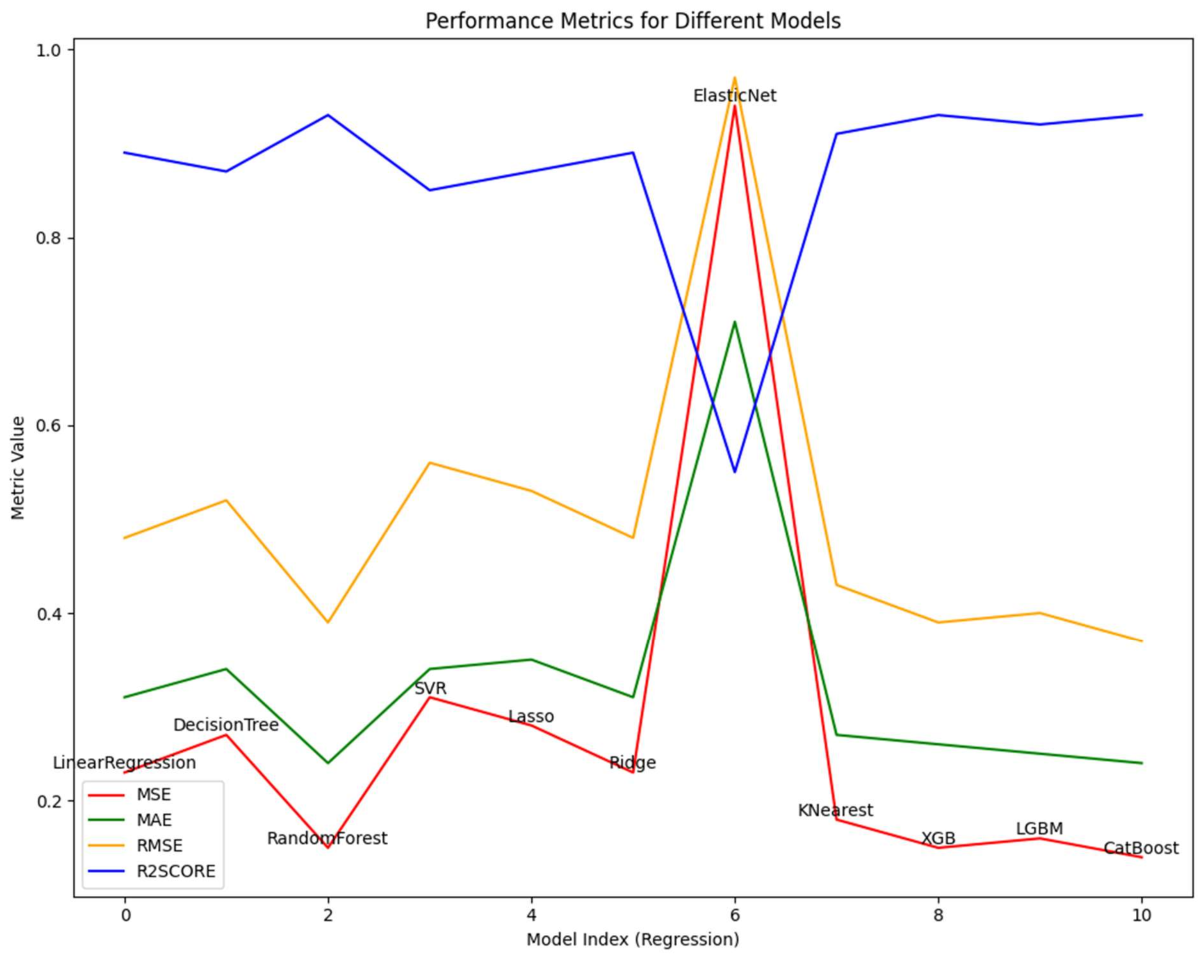
- KNeighborsRegressor:
  - MSE: 1.41
  - MAE: 0.95
  - RMSE: 1.19
  - R2SCORE: 0.70

- XGBRegressor:

  - MSE: 1.34
  - MAE: 0.86
  - RMSE: 1.16
  - R2SCORE: 0.72

- LGBMRegressor:
  - MSE: 1.20
  - MAE: 0.82
  - RMSE: 1.10
  - R2SCORE: 0.75

- CatBoostRegressor:
  - MSE: 1.25
  - MAE: 0.85
  - RMSE: 1.12
  - R2SCORE: 0.74

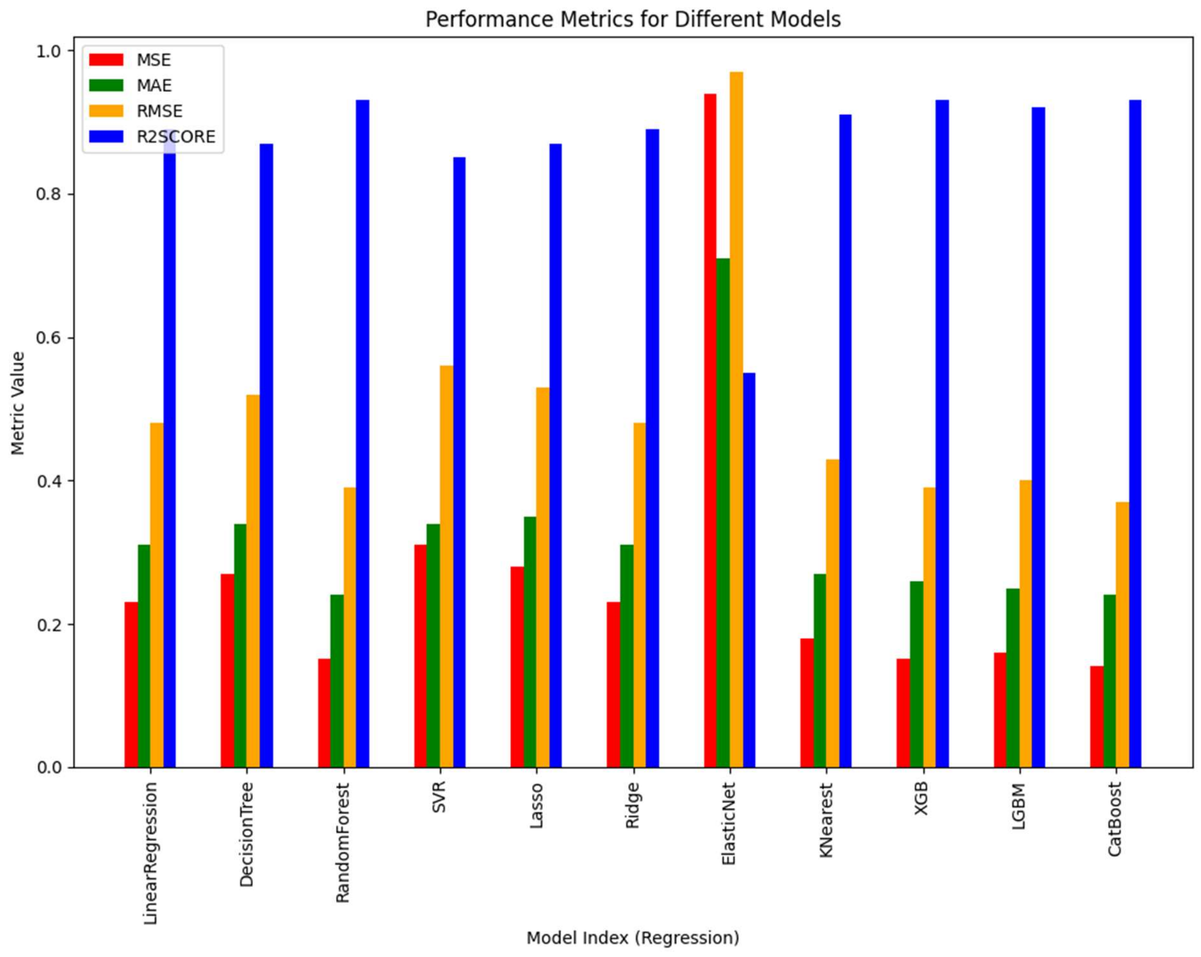


From the evaluation of various regression models on the air quality dataset, it can be observed that the Random Forest Regressor and LGBMRegressor perform the best among the tested models, achieving lower values of mean squared error (MSE), mean absolute error (MAE), and root mean squared error (RMSE). These models also demonstrate higher values of the R2 score, indicating a better fit to the data.

On the other hand, models such as DecisionTreeRegressor and SVR show higher values of MSE and RMSE, suggesting that they have relatively higher prediction errors compared to the better-performing models.

The results also indicate that linear models such as LinearRegression, Lasso, and Ridge perform comparably to each other, achieving similar values for the evaluated metrics. This suggests that the relationships between the input features and the target variable may not be strictly linear.

Overall, the Random Forest Regressor and LGBMRegressor are recommended for predicting carbon monoxide levels in air quality based on their superior performance in terms of accuracy and predictive power. These models can be further fine-tuned and optimized to achieve even better results.



FUTURE WORK:

It is worth mentioning that the performance of the models may vary depending on the specific air quality dataset and the target variable being predicted. Further analysis and experimentation with different models, feature engineering techniques, and hyperparameter tuning could potentially lead to even better results and more accurate predictions.

In conclusion, the air quality dataset provides valuable information for predicting air pollutant levels. The regression models applied to the dataset demonstrate their potential to estimate carbon monoxide levels accurately. This information can be utilized for monitoring and managing air quality, enabling proactive measures to mitigate the adverse effects of air pollution on human health and the environment.

REEFERENCES:

• Air Quality Data Set. UCI Machine Learning Repository. Retrieved from https://archive.ics.uci.edu/dataset/360/air+quality

• Pandas Documentation. Retrieved from https://pandas.pydata.org/docs/

• Scikit-learn Documentation. Retrieved from: https://scikit-learn.org/stable/documentation.html

• Seaborn Documentation. Retrieved from https://seaborn.pydata.org/

• Matplotlib Documentation. Retrieved from https://matplotlib.org/stable/contents.html

• SimpleImputer Documentation. Retrieved from : https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html

• Train-Test Split Documentation. Retrieved from: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

• RobustScaler Documentation. Retrieved from: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

• Regression Models in Scikit-learn. Retrieved from https://scikit-learn.org/stable/supervised_learning.html#regression