Technical Implementation

Team Information

• Team Member : Pavan Sai Porapu

• Contact Email: porapupavansai.22.csm@anits.edu.in

1. Overview of Technical Implementation

▶ What is the core functionality of your solution?

- The core functionality of this solution is to estimate **Air Quality Index (AQI)** based on various environmental and meteorological factors.
- The system uses past three days air quality index data based on factors such as concentration of pollutants (SO2,O3,NO2,CO ect.) of metropolitan cities in india for training.
- The data is pre-processed using,
 - Numerical features scaling (StandardScaling, MinMaxScaling)
 - ➤ Categorical features OneHot Encoding
 - Feature expansion Polynomial feature expansion of degree 2
- Different traditional and statistical machine learning models are trained to accurately predict the air quality index at a given location.

➤ How does your model integrate with the API data?

- **Training Phase:** The model is trained using historical data collected from weather APIs (Rapidapi.com)
- **Prediction Phase:** The system does not fetch real-time data but instead relies on user-inputted or pre-stored data to estimate AQI.

➤ What machine learning or deep learning techniques are used?

Trained multiple models such as both traditional ML and ensemble learning.

- Regression Models: Linear Regression, Ridge
- Tree-Based Models: Decision Tree, Random Forest, Gradient Boosting
- Advanced Boosting Models: XGBoost
- Stacking Ensemble: Random Forest, Decision Tree, and Gradient Boosting.
- **Hyperparameter Tuning:** GridSearchCV

Used Ensembled technique to get more accurate results. Stacking random forest regressor, Decision tree regressor and gradient boosting as final predictor.

2. ML Project Architecture

➤ What are the key stages in the end-to-end ML workflow?

(Describe the entire process from data collection to model deployment.)

End-to-End ML workflow:

1. Data Collection

- o Historical air pollution data are collected from weather APIs.
- o Data includes pollutant levels (PM2.5, PM10, NO2, CO, etc.) & location.

2. Data Preprocessing

- o Handle missing values by imputation or removal.
- Convert timestamps to datetime format and extract relevant features (e.g., time of day, season).
- o Normalize data to a certain range to ensure consistency.

3. Feature Engineering

- o Generate polynomial features to capture non-linear relationships.
- o Encode categorical data if necessary.
- o Understanding Spread of data using Visualizations.

4. Model Training

- o Train various models (Linear Regression, Decision Trees, Random Forest, Gradient Boosting, XGBoost, etc.).
- o Use cross-validation (5-fold) to evaluate generalization performance.
- o Perform hyperparameter tuning using GridSearchCV.

5. Model Evaluation

- Use metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R² score.
- o Compare models and choose the best-performing one.

6. Model Deployment

- o Model is integrated in a web- interface and deployed (Streamlit)
- 7. Monitoring & Maintenance (Optional for Future Scalability)
 - o Expanding model's knowledge by training various locations data.

➤ How does data flow through the system?

(Explain how raw data is collected, processed, stored, and used for training.)

1. Data Collection

- Data is collected from an api endpoint and stored in .csv file format.
- Data includes pollutants (PM2.5, PM10, NO2, etc.).
- 2. Data Processing & Storage
 - Handle missing values, convert timestamps, and normalize using StandardScaler.
 - Store cleaned data in CSV files for training.
- 3. Model Training
 - Data is split into training and testing sets.
 - Polynomial features are generated, and models are trained using regression and ensemble techniques.

4. Prediction & Output

• The trained models is saved and later used with user-inputted data for AQI prediction.

> What are the main components of the ML pipeline?

(Break down data acquisition, preprocessing, feature engineering, model training, evaluation, and deployment.)

- Data Acquisition Collect historical air pollutant data from APIs.
- Preprocessing Clean data, handle missing values, normalize, and transform features.
- Feature Engineering Generate polynomial features and select key variables.
- Model Training Train multiple models (Random Forest, XGBoost, etc.) with hyperparameter tuning.
- Evaluation Measure performance using RMSE, R², and cross-validation.
- Deployment Save the best model for making AQI predictions from user inputs.

3. Model Selection & Justification

Which model(s) did you choose and why? (e.g., Logistic Regression, SVM, Random Forest, Neural Networks, etc.)

- 1. Trained multiple regression models randomly and based on observations of the evaluation metrices and performance of the models, decided to use an ensemble learning technique.
- 2. We stacked three models Gradient Boosting, Decision Tree regressor, Random Forest regressor, obtained an accurate model with R-squared score of 0.98 on both training and testing data.

Were multiple models tested? If yes, what comparisons were made?

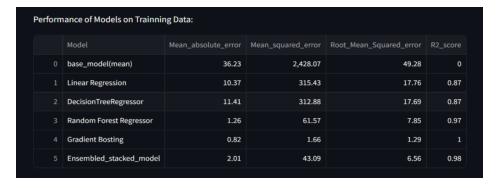


Figure 1:Evaluation metrics of models on training data



Figure 2: Evaluation metrics of models on testing data

Was transfer learning or pre-trained models used?

No Transfer learning or pre-trained models are applicable.

4. Training Process

- ➤ How was the data split for training, validation, and testing?
 - The data is divided into two splits
 - 1. Testing data -- 80%
 - 2. Training data 20%
 - This is Achieved by using scikit-learn library using train test split function
- ➤ What training techniques were used (e.g., data augmentation, early stopping)?
 - Hyperparameter tuning Gridsearchev
 - Regularisation to prevent overfitting to training data
 - Ensembled learning technique
- ➤ What loss function and optimization algorithms were used (e.g., Adam, SGD)?
 - Loss Function: Mean Squared Error (MSE) for regression models.
 - Optimization Algorithms:
 - 1. Gradient Descent variants (used in models like Ridge, Lasso, ElasticNet).
 - 2. Boosting algorithms (XGBoost, Gradient Boosting) use custom tree-based optimization.
- > Did you apply hyperparameter tuning? If so, what method was used (GridSearch, RandomSearch, Bayesian Optimization)?
 - Yes, applied.
 - Method Used: GridSearchCV for exhaustive search over parameter combinations.
 - Models Tuned: Random Forest, Gradient Boosting (learning rate, depth, estimators).

5. Model Evaluation & Performance Metrics

➤ What performance metrics were used for evaluation (e.g., Accuracy, Precision, Recall, F1 Score, RMSE)?

Since this is a **regression problem**, the following metrics were used:

- Mean Absolute Error(MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R² Score

➤ How does your model compare to a baseline model?

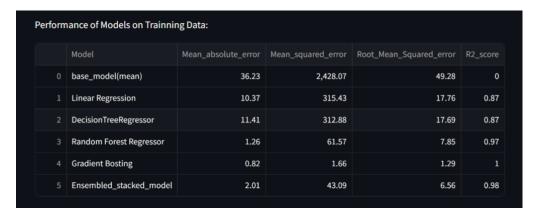


Figure 3: Comparison of basedline model and other ml models

➤ What were the key takeaways from the evaluation?

- 1. Ensemble Models Performed Best
 - Stacked models (Random Forest + Gradient Boosting) outperformed individual models.
- 2. Feature Engineering Improved Accuracy
 - Polynomial features helped capture complex relationships, improving performance.
- 3. Hyperparameter Tuning was Essential
 - GridSearchCV optimized parameters, significantly reducing RMSE.
- 4. MAE & RMSE Showed Low Prediction Errors
 - The model made reasonably accurate predictions with minimal large deviations.

6. Optimization & Engineering Enhancements

Performance Optimization Techniques

- ➤ Was model compression or pruning used?
 - Yes Used, techniques like tree pruning (for Decision Trees & Random Forests)
- > Did you implement quantization or reduced precision techniques?
 - Not implemented, but could improve inference speed.
- ➤ How was inference speed optimized?
 - Used StandardScaler for fast data transformation.
 - Stacking Regressor optimized predictions by combining multiple models.

Code Efficiency & Engineering Best Practices

> Was the model deployed in an optimized environment (e.g., GPU acceleration, parallel processing)?

Model is going to be deployed in a free version in github or huggingface spaces.

- What libraries or frameworks were used for performance improvements?
 - Scikit-learn, XGBoost for efficient ML operations.
 - Pandas & NumPy for fast data handling.
 - Streamlit for developing user- friendly interface.
- ➤ How was code structured for maintainability and scalability?
 - Modular approach (separate data processing, training, and evaluation steps).

7. Model Deployment & Integration

- ➤ Was the model deployed? (e.g., Flask API, FastAPI, Streamlit, Cloud services) Provide the link to access your project
 - No the model is not deployed yet
- **➤** How does the model interact with the user interface or API?
 - The model interact with the user through an interface developed using streamlit
- > What challenges were faced during deployment and how were they resolved?

8. Challenges & Solutions

- **▶** What were the biggest technical challenges faced during implementation?
 - Selecting the Best Model -- Balancing accuracy & inference speed was difficult.
 - Hyperparameter Tuning Complexity -- Finding optimal settings for models like XGBoost took time.
 - Time constraint Due to lack of time I skipped some of the things which are may be essential.
- **➤** How did your team overcome them?
 - Model Evaluation & Stacking Combined models to improve accuracy.
 - GridSearchCV for Hyperparameter Tuning Optimized parameters efficiently.
 - Prioritize the tasks at hand to complete due to time constraint.

- 9. Supporting Code & References (If applicable)
 - > Attach or provide links to code snippets showcasing technical implementation.
 - > Mention any references or papers that supported your implementation.