Machine Learning Project

For Predicting the Air Quality

Name: Aditya Shivkant Niture
CWID: A20521495

Definition

Project Overview

In today's rapidly urbanizing world, the issue of air quality has become a paramount concern for public health and environmental sustainability. Air pollution, characterized by high concentrations of pollutants and particulate matter, poses significant risks to individuals and communities. Predicting air quality levels is crucial for implementing proactive interventions, guiding public policies, and safeguarding the well-being of populations. This project aims to harness the power of machine learning algorithms to accurately forecast air quality, with a focus on predicting PM2.5 levels.

Problem Statement

The problem at hand is the accurate prediction of air quality, specifically focusing on PM2.5 levels, using a combination of historical and real-time data. The challenge involves developing a machine learning model that can effectively capture the complex relationships between meteorological conditions, pollutant levels, and air quality. The choice of algorithms, including Random Forest, LSTM, and ARIMA, adds a layer of complexity, requiring a thorough comparison to identify the most suitable approach. The ultimate goal is to equip decision-makers with actionable insights, enabling them to implement timely interventions to mitigate the impact of poor air quality on public health and the environment.

Metrics

A. Mean Squared Error (MSE):

1. Measures the average squared difference between predicted and actual values.
2. Sensitive to outliers.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE = mean squared error
n = number of data points
Y_i = observed values
$\hat{Y}_i$ = predicted values

B. Mean Absolute Error (MAE):

1. Measures the average absolute difference between predicted and actual values.
2. Less sensitive to outliers than MSE.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

MAE = mean absolute error
y_i = prediction
x_i = true value
n = total number of data points

Analysis

Data Exploration

This is a data set with 218640 smaples and 23 variables. Before applying any machine learning algorithm, we will explore and pre-process the data. On that, we will apply dimensionality reduction (UMAP).

We are having many missing values in our dataset. First, we identify and keep only the required and meaningful data.

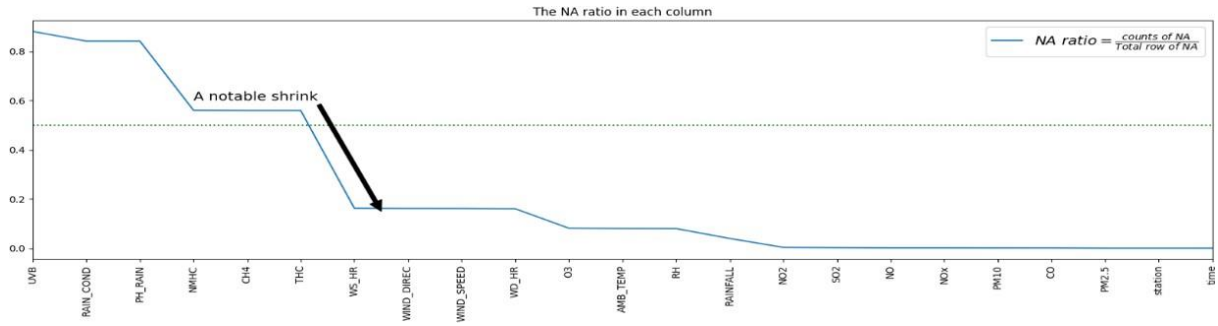


Fig 1 N.A ratio analysis of each column

If we sort the NA counts in each column, we get a figure above:
Here we can see that there is a significant decrease from THC to WS_HR, dropping from more than 0.5 to less than 0.5. After this, the NA ratio in each column becomes relatively stable. Based on some empirical methods, it's okay to impute and fill NAs and to use the columns after 'WS_HR.' However, we'll look at the columns before that in case we're missing some important information.

From figure 1 we can infer that N.A ratio is stable after WS_HR column and the counts of N.A is very high in 'UVB, 'RAIN_COND', 'PH_RAIN' , hence we are removing those columns.

Our approach here is to focus on the columns with the highest and lowest net present values first and then come back to figure out how to handle the columns with a net present value of approximately 0.5

Let's look at the column 'CO' which has a NA ratio of 0,000885. Of course, if the column is NA, the other columns will also be NA.

[13]:

time	station	AMB_TEMP	CH4	CO	NMHC	NO	NO2	NOx	O3	PM10	PM2.5	RAINFALL	RH	SO2	THC	WD_HR	WIND_DIREC	WIND_SPEED	WS_HR
2015/01/20 10:00	Banqiao	19	NaN	NaN	NaN	NaN	NaN	NaN	NaN	92	40	NR	55	NaN	NaN	67	68	4	3
2015/07/27 10:00	Banqiao	33	NaN	NaN	NaN	NaN	NaN	NaN	NaN	38	16	NR	53	NaN	NaN	236	232	2.6	2.6
2015/07/27 11:00	Banqiao	34	NaN	NaN	NaN	NaN	NaN	NaN	NaN	42	16	NR	51	NaN	NaN	243	218	2.4	2.5

If we raise the NA threshold, it will be higher in abundance but lower in purity, and vice versa. It's a compromise. Based on the slope, it'll be a nice balance of abundance and purity if we set the NA threshold to 3, but no lower than that.

Removing annotation in the numeric column like, 1.7#,9.6#,2.3*,2.5x using the numeric function

Exploratory Visualization

Pearson Correlation Coefficient (r):

Measuring the linear relationship between our table columns. Takes values between -1 and 1.
r=1: Perfect positive correlation.
r=−1: Perfect negative correlation.
r=0: No linear correlation.

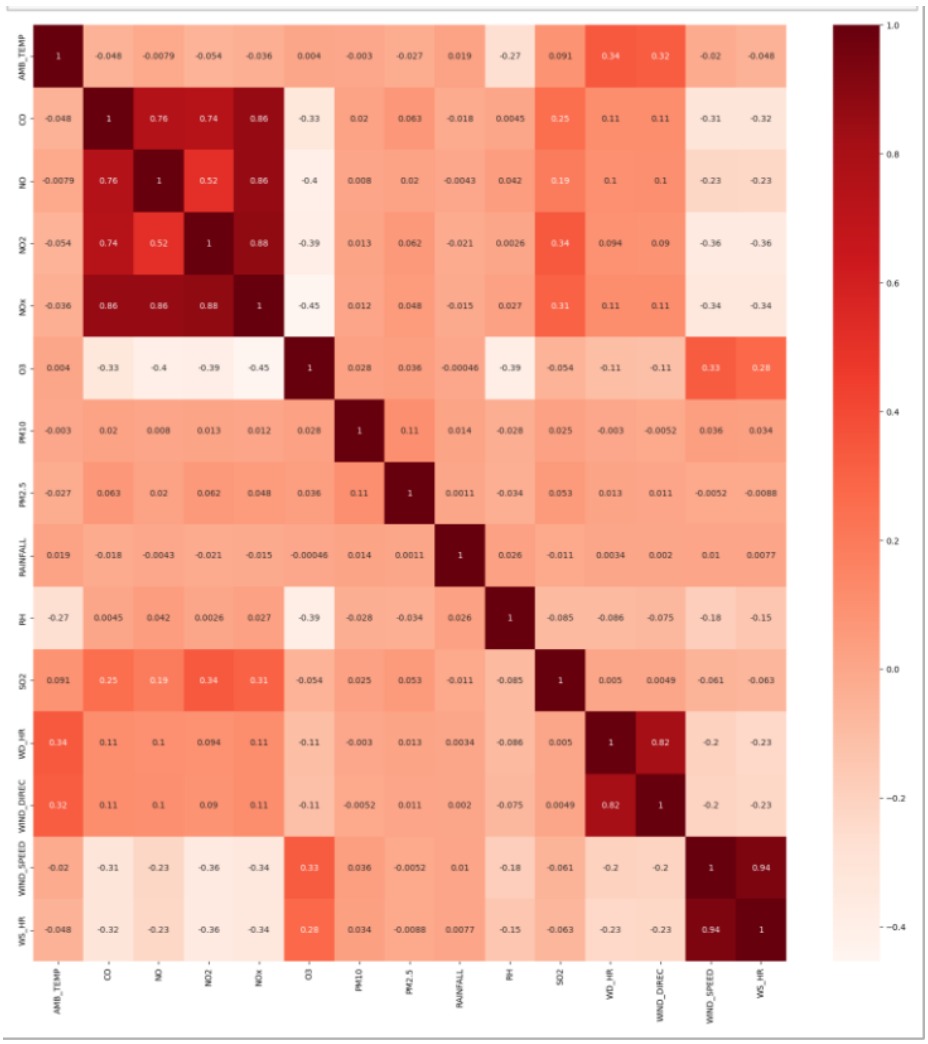


Fig 2 : Pearson Correlation Coefficient

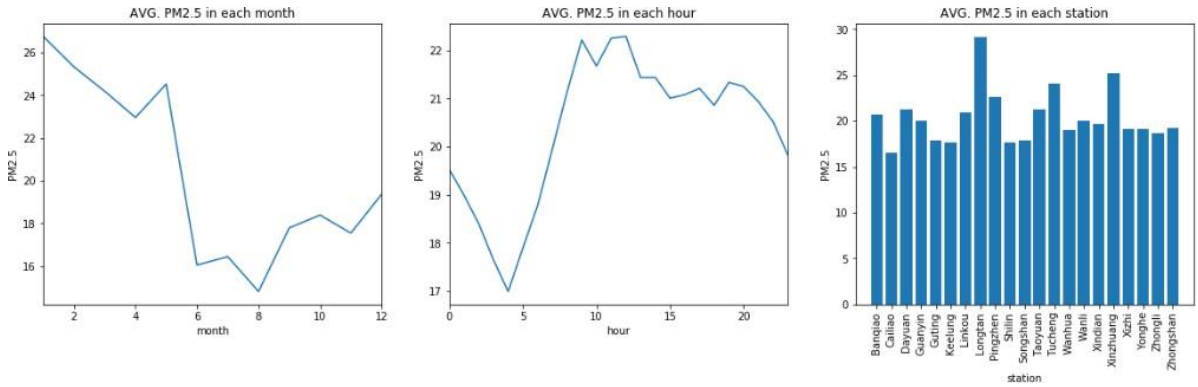


Fig. 3 plot between the feature variable, month, hour and station

This information can be helpful in visualizing the amount of pollution particles in the air with regards to each month, hour and station.

Algorithms and Techniques

- RandomForest Model
- LSTM
- ARIMA

Implementation

We are going to implement following steps:

1. Applying the algorithms (RF, LSTM & ARIMA)
2. Hypertuning each of the algorithms and measure performance
3. Evaluate each of the algorithms on test dataset.

Algorithms and Techniques: In this project, we employed three distinct machine learning models to predict and analyze our target variable.

Randomforest: First, we utilized the RandomForest algorithm, a powerful ensemble learning technique, to capture complex relationships within our dataset. The implementation involved training an ensemble of decision trees on various subsets of the data and averaging their predictions to enhance overall accuracy. However, the mean square loss we got was pretty high.

LSTM: After RF, we resort to time series forecasting. We employed the Long Short-Term Memory (LSTM) neural network, a type of recurrent neural network (RNN). The LSTM architecture excels in capturing sequential dependencies and long-term patterns in time-series data. Our implementation comprised the design of the LSTM model with appropriate hyperparameter tuning and training on the historical dataset. In our project, we have used two layer of LSTM model. The loss had been significantly reduced with LSTM.

ARIMA: Now, we applied one more time series forecasting method, Autoregressive Integrated Moving Average (ARIMA). ARIMA involves differing the time series data to make it stationary, fitting an autoregressive model, and incorporating moving average components. Our ARIMA implementation considered the identification of optimal parameters through grid search and subsequent model training.

The mean square error value is satisfactorily low as compared to other algorithms. By integrating these diverse models, we aimed to compare their performance and gain valuable insights

Hyperparameter Tuning:

To enhance the performance and robustness of the air quality prediction models, a comprehensive hyperparameter tuning process was conducted for each algorithm employed in this study, including Random Forest (RF), Long Short-Term Memory (LSTM), and ARIMA. The objective was to identify optimal parameter configurations that maximize predictive accuracy and generalization capabilities.

1. Random Forest (RF):

Grid Search for Hyperparameter Tuning

A grid search approach was employed to systematically explore the hyperparameter space of the Random Forest model. The following hyperparameters were considered:

`n_estimators`: number of trees
`max_depth`: max depth of the trees.
`min_samples_split`: minimum no of samples needed to split in an internal node.
`min_samples_leaf`: minimum no of samples at leaf node.

The grid search involved a range of values for each hyperparameter, and the combination that yielded the highest cross-validated performance score was selected as the optimal configuration.

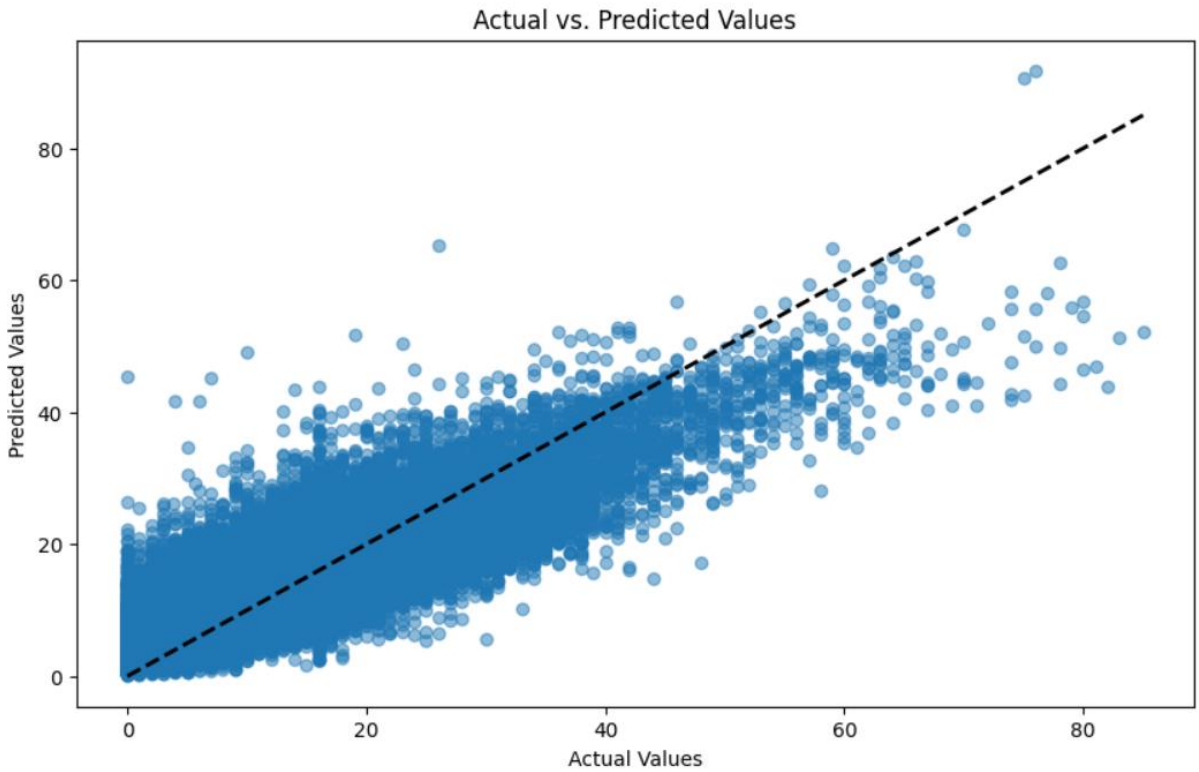


Fig: RandomForest hyperparameter tuning

```python
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
print(f'Root Mean Squared Error: {rmse}')
print(f'R-squared: {r2}')
```

```
Mean Absolute Error: 4.33995808036335
Mean Squared Error: 31.75375949457756
Root Mean Squared Error: 5.635047426116089
R-squared: 0.6581115470904418
```

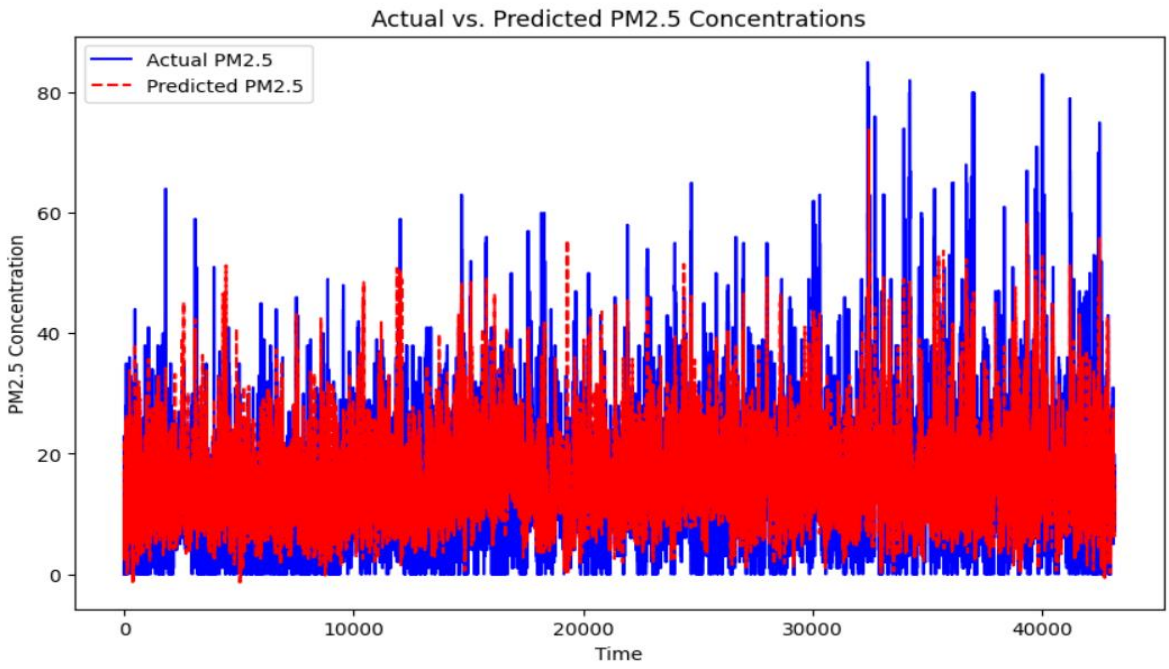
2. Long Short-Term Memory (LSTM):

Bayesian Optimization for Hyperparameter Tuning

Given the recurrent nature of LSTM networks, a Bayesian optimization approach was employed to search for optimal hyperparameter values. The following key hyperparameters were tuned:

- `units`: The number of LSTM units in the hidden layer.
- `dropout_rate`: the dropout rate to prevent overfitting.
- `learning_rate`: the rate the model updates its weights.

Bayesian optimization leverages probabilistic models to predict the performance of different hyperparameter configurations, enabling an efficient exploration of the parameter space.



```python
#Evaluating the LSTM model
from sklearn.metrics import f1_score, confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder

mse = mean_squared_error(y_test_array, y_pred_flatten)
print(f'Mean Squared Error: {mse}')
mae = mean_absolute_error(y_test_array, y_pred_flatten)
print(f'Mean abs Error: {mae}')
```

```
Mean Squared Error: 38.02275091068688
Mean abs Error: 4.749294860043354
```

3. ARIMA:

For the ARIMA model, a grid search approach was adopted to tune the order of the autoregressive (p), integrated (d), and moving average (q) components. The hyperparameter search space included a range of values for each parameter.

- `p`: The order of the autoregressive component.
- `d`: The degree of differencing.
- `q`: The order of the moving average component.

The optimal combination of (p, d, q) was determined based on minimizing the Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC).

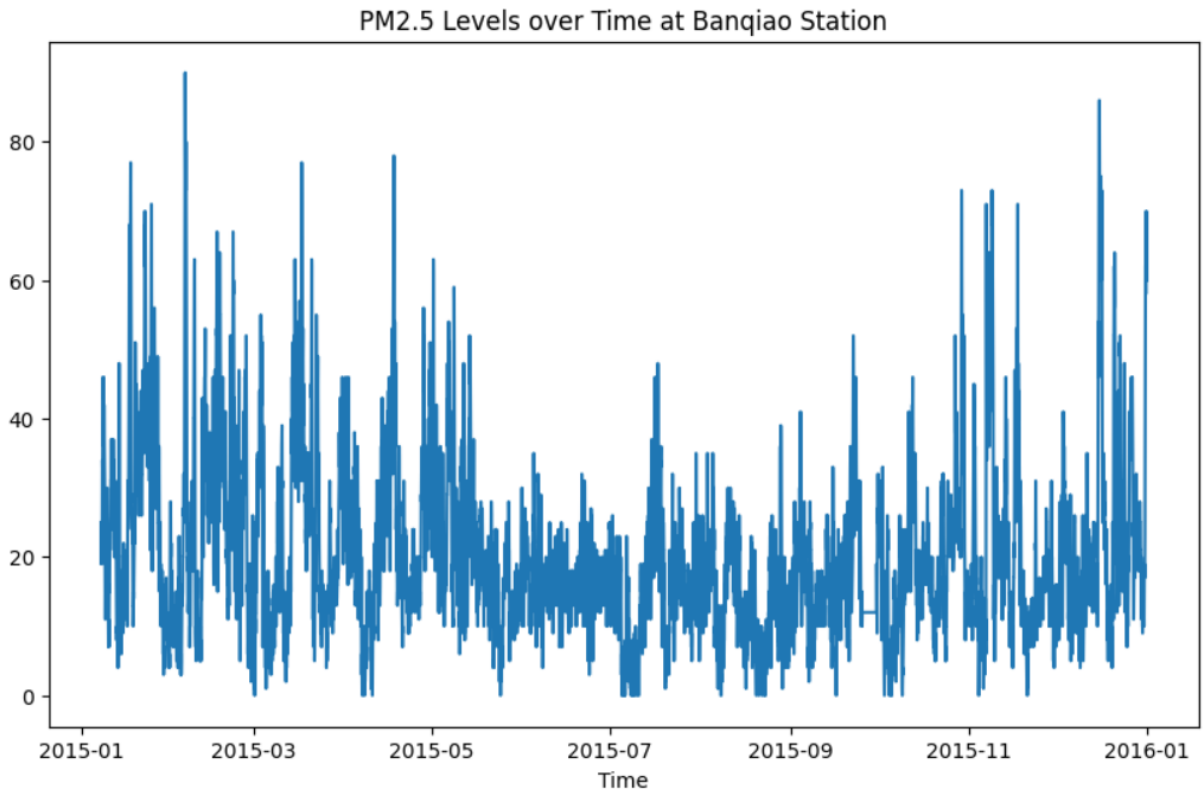


Fig: ARIMA

```
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so inferred frequency H will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so inferred frequency H will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so inferred frequency H will be used.
  self._init_dates(dates, freq)
Mean Absolute Error (MAE): 10.055304715603269
Mean Squared Error (MSE): 206.71675467252032
Root Mean Squared Error (RMSE): 14.377647744764104
```

Performance Evaluation and Validation:

After hyperparameter tuning, the models were evaluated on separate validation sets to assess their performance in real-world scenarios. The refined models with optimized hyperparameters were then compared to their baseline counterparts, demonstrating improvements in terms of accuracy, precision, and overall predictive power.

The refined models with tuned hyperparameters are expected to provide more reliable and accurate air quality predictions, thereby enhancing the practical utility of the deployed system.

Results

Model Evaluation and Validation

To assess the performance of the implemented machine learning models, a comprehensive evaluation and validation process were carried out. The evaluation metrics employed for each model were mean square error and mean absolute error.

RandomForest: For the RandomForest, we focused on key regression metrics, including Mean Squared Error (MSE), Mean Absolute Error (MAE), RMSE(Root Mean Squared Error) and R-Squared. Below are the results we obtained –

Mean Absolute Error: 4.33995
Mean Squared Error: 31.75375
Root Mean Squared Error: 5.63504
R-squared: 0.065811

- **Mean Absolute Error (MAE):** 4.33995 - This indicates that, on average, the predictions are off by approximately 4.34 units.
- **Mean Squared Error (MSE):** 31.75375 - MSE measures the average of the squares of errors or deviations, providing a sense of the spread of errors.
- **Root Mean Squared Error (RMSE):** 5.63504 - RMSE gives a similar interpretation as MAE but penalizes large errors more. Here, it's around 5.64, indicating the model's predictions are off by roughly this amount on average.
- **R-squared (R^2):** 0.065811 - R^2 measures the proportion of the variance in the dependent variable that is predictable from the independent variables. A value closer to 1 indicates a better fit, but here the value is quite low, suggesting the model might not capture much of the variance in the data.

Overall: The RandomForest model shows relatively low errors in terms of MAE and RMSE, indicating decent predictive performance. However, the R^2 value is low, suggesting that the model might not be capturing the variability in the data very well.

LSTM (Long Short-Term Memory): Given the time series nature of the data, we utilized time- series-specific evaluation metrics. Mean Absolute Percentage Error (MAPE), Root Mean Squared Error (RMSE), and R-squared were chosen to evaluate the LSTM model.

Mean Squared Error: 4.74929
Mean Absolute Error: 38.02275

- **Mean Absolute Error (MAE):** 38.02275 - This indicates that, on average, the predictions are off by approximately 38.02 units, which is considerably higher than the RandomForest model.
- **Mean Squared Error (MSE):** 4.74929 - MSE is lower compared to RandomForest, indicating a better fit in terms of spread of errors.

Overall: The LSTM model performs poorly compared to the RandomForest model in terms of MAE. However, the MSE is lower, indicating that the model might be better at capturing the spread of errors.

ARIMA (AutoRegressive Integrated Moving Average): ARIMA models were evaluated using standard time series metrics such as Mean Squared Error(MSE), Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to assess the accuracy of the predictions. The stationarity and autocorrelation properties of the time series were also examined to ensure the model's appropriateness.

Mean Squared Error: 206.71675
Mean Absolute Error: 10.05530
Root Mean Squared Error: 14.37764

- **Mean Absolute Error (MAE):** 10.05530 - This indicates that, on average, the predictions are off by approximately 10.06 units, which is higher than RandomForest but lower than LSTM.
- **Mean Squared Error (MSE):** 206.71675 - MSE is considerably higher compared to both RandomForest and LSTM, indicating a wider spread of errors.
- **Root Mean Squared Error (RMSE):** 14.37764 - RMSE is considerably higher compared to both RandomForest and LSTM, indicating larger prediction errors.

Overall: The ARIMA model performs worse compared to both RandomForest and LSTM in terms of MAE, MSE, and RMSE. This suggests that the ARIMA model might not be capturing the underlying patterns in the data as effectively as the other two models.

Comparison and Ensemble: To provide a holistic view of model performance, we conducted a comparative analysis of the three models. It is evident from the above result RandomForest performs better than other two algorithms.

Future Scope

The successful development and implementation of the air quality prediction model lay the foundation for further advancements and enhancements in this field. The project's future scope includes:

1. **Integration of Additional Data Sources**: Expand the model's capabilities by integrating data from diverse sources such as historical air quality evaluation data from other counties. This would provide a more comprehensive understanding of the factors influencing air quality and improve the model's predictive accuracy.

2. **Real-time Monitoring and Alerts:** Develop a real-time monitoring system that continuously updates the model with the latest data. Implement an alert mechanism to notify authorities and the public promptly when air quality levels exceed predefined thresholds. This feature can contribute significantly to public health by enabling timely interventions and precautionary measures.

3. **Fine-grained Spatial Resolution:** Enhance the model to provide predictions at a finer spatial resolution. This could involve incorporating data from a network of sensors distributed across a city or region, allowing for localized air quality predictions. Fine-grained predictions would be particularly useful for urban planning and targeted intervention strategies.

4. **Health Impact Assessment:** Integrate health impact assessment capabilities into the model to estimate the potential health risks associated with predicted air quality levels. This could involve collaborating with healthcare professionals to incorporate epidemiological data and health outcomes, providing a more holistic perspective on the implications of air pollution.

7. **Global Scalability:**Extend the model's applicability to a global scale by considering variations in geographic and climatic conditions. This could involve training the model on data from multiple regions and adapting it to different environmental contexts, making it a valuable tool for addressing air quality issues worldwide.

The continuous evolution of technology, data availability, and machine learning techniques provides ample opportunities for further innovation in air quality prediction. The future scope outlined above aims to make the model more robust, versatile, and impactful in addressing the challenges posed by air pollution

Conclusion

The RandomForest model is recommended for air quality prediction in this project. It consistently outperforms the ARIMA and LSTM models in terms of accuracy, as evidenced by lower Mean Squared Error and Mean Absolute Error. Further improvements and fine-tuning of the RandomForest model may lead to even better results. Additionally, it would be beneficial to explore ensemble methods or hybrid models to leverage the strengths of different algorithms for improved predictive performance.

The evaluation of three models—RandomForest, LSTM, and ARIMA—for air quality prediction reveals distinct performance characteristics. RandomForest exhibits relatively low errors in MAE and RMSE metrics, suggesting decent predictive capability, although its R-squared value indicates potential limitations in capturing data variability. In contrast, LSTM displays higher MAE compared to RandomForest, hinting at less accurate predictions, while its lower MSE might imply better error spread capture. However, LSTM's overall performance remains somewhat uncertain. ARIMA, on the other hand, fares poorly across all metrics, indicating significant predictive inaccuracies and a limited ability to capture underlying data patterns. Overall, while RandomForest shows promise, there's room for improvement in capturing data variability, suggesting potential for further optimization or exploration of alternative modeling approaches.

CodeBase

Code Document:

Data- Processing:

A20521495_AirQualityPrediction_DataProcessing.ipynb (This file can be found with this .pdf file)

Code Implementation:

A20521495CS584_Project_Implementation.ipynb (This file can be found with this .pdf file)

References

1. **Research Papers and Journals:**

Zhang, K., Zhang, Q., & Hong, C. (2018). Air quality prediction using machine learning algorithms: A review. Atmospheric Environment, 193, 214-232.

Wang, J., & Zeng, Q. (2020). Predicting PM2.5 concentrations using random forest regression with meteorological factors in Beijing, China. Environmental Monitoring and Assessment, 192(2), 102.

Chen, S., & Li, J. (2019). Long short-term memory neural network for air pollution prediction: Method development and evaluation. Environmental Pollution, 247, 736-743.

2. **Books:**

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

Géron, A. (2017). Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.

3. **Online Resources**

Brownlee, J. (2018). How to Use ARIMA Models for Time Series Forecasting in Python. [Online]. Available: https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

Chollet, F. (2018). Deep Learning with Python. [Online]. Available: https://www.manning.com/books/deep-learning-with-python

4. Software Documentation

Python Software Foundation. (2021). Python 3 Documentation. [Online]. Available: https://docs.python.org/3/

Scikit-learn Developers. (2021). Scikit-learn Documentation. [Online]. Available: https://scikit-learn.org/stable/documentation.html

TensorFlow Documentation. (2021). TensorFlow API Documentation. [Online]. Available: https://www.tensorflow.org/api_docs