Multi-variate Time Series Forecasting - Air Quality

Team members: Saiabhinav Chekka, Nishant Pagadala, Preetham Vinnamala, Sai Srikar Gandhe

ENGR-E503 : INTRO TO INTELLIGENT SYSTEMS

***Abstract*—This report presents a detailed analysis of multi-variate time series forecasting applied to air quality prediction, which is a critical aspect of environmental monitoring and to control the pollution. With a large-scale air quality dataset which we acquired from UCI Machine Learning repository, we carefully investigated and contrasted a variety of sophisticated forecasting techniques, such as VAR, SARIMAX, FB Prophet, LSTM (concentrating on a stacked bidirectional architecture), Ensemble approaches, TCN, and GRU models. In order to maximise performance based on the distinct features of the dataset, we perfromed hyperparameter tuning on each model. To evaluate each model's accuracy and precision, we used a rigorous evaluation methodology that was centred on the Root Mean Squared Error (RMSE) statistic. We observed that the stacked bidirectional LSTM model significantly outperformed the other models, delivering the highest accuracy with the lowest RMSE. In addition to quantitative evaluations, we also plotted the forecasting graphs that visualize the predictive power of each algorithm using which multiple insights could be drawn. This paper shows how sophisticated contemporary forecasting methods can be and how important they are to environmental data science.**

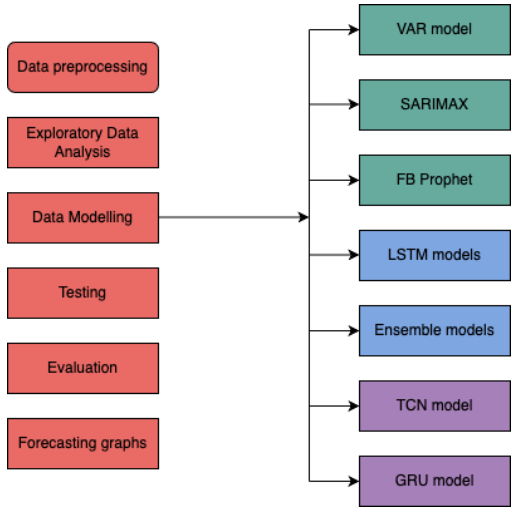
I. Introduction

We believe that air quality poses a major risk to public health and the environment due to its volatility and unpredictability, which makes population health and urban development management challenging. Since prevention is always preferable to cure, the advancement of deep learning and machine intelligence has opened up new avenues for accurate air quality index (AQI) forecasting as well as the opportunity to address health concerns related to pollution.

Several machine learning and deep learning models are used in our project, "Multi-variate Time Series Forecasting - Air Quality," to accurately predict air quality indices. We conduct a systematic assessment of the performance of several algorithms, such as VAR, SARIMAX, FB Prophet, LSTM, TCN, and GRU. We pay special attention to a stacked bidirectional LSTM model that is well-known for its ability to handle temporal data. The study includes a thorough curation of data, pre-processing, hyperparameter tuning for model optimisation, evaluation using the Root Mean Squared Error metric, and plotting of forecasting graphs for future dates. This introduction lays the groundwork for a thorough examination of the models' predictive capabilities and real-world uses..

II. Methods

The following sections represent the way in which the multi-variate time series forecasting was done.



Flow chart of methodology

A. *Data preprocessing*

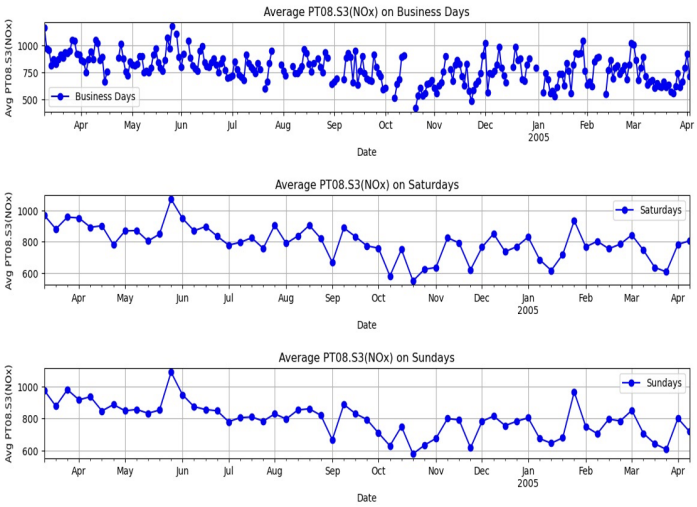
Our data preprocessing involved cleaning and readying the dataset for modeling. We removed two irrelevant columns and discarded the last 114 records full of NaN values to enhance data integrity. Additionally, we standardized numerical values by converting commas to decimal points and merged 'Date' and 'Time' into one column for better time series analysis.

Subsequently, we addressed error values of negative sensor readings by imputing missing data with column means or medians, except for the NHMC(GT) column, which was removed due to excessive missing values. Outliers were corrected using the Interquartile Range method, ensuring a reliable dataset for accurate forecasting models.

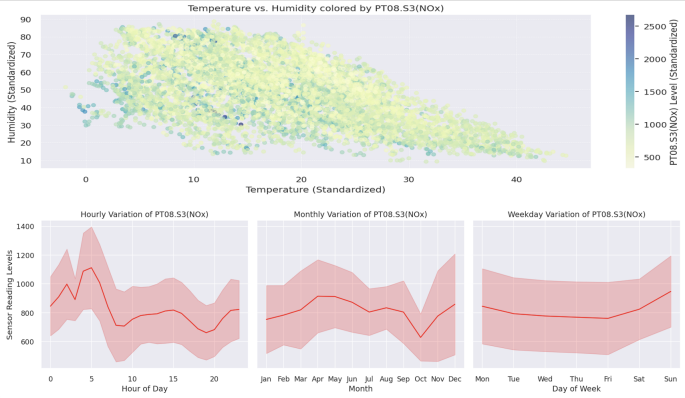
	DateTime	CO(GT)	PT08.S1(CO)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08.S5(O3)	T	RH	AH
0	2004-03-10 18:00:00	2.6	1360.0	11.9	1046.0	166.0	1056.0	113.0	1692.0	1268.0	13.6	48.9	0.7578
1	2004-03-10 19:00:00	2.0	1292.0	9.4	955.0	103.0	1174.0	92.0	1559.0	972.0	13.3	47.7	0.7255
2	2004-03-10 20:00:00	2.2	1402.0	9.0	939.0	131.0	1140.0	114.0	1555.0	1074.0	11.9	54.0	0.7502
3	2004-03-10 21:00:00	2.2	1376.0	9.2	948.0	172.0	1092.0	122.0	1584.0	1203.0	11.0	60.0	0.7867
4	2004-03-10 22:00:00	1.6	1272.0	6.5	836.0	131.0	1205.0	116.0	1490.0	1110.0	11.2	59.6	0.7888

Dataset after cleaning

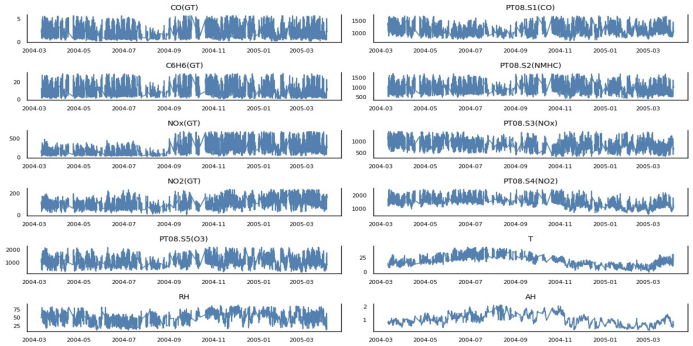
B. *Exploratory Data Analysis*



The provided plots compare PT08.S3(NOx) levels across business days and weekends, illustrating temporal pollution trends and enabling a concise comparison of weekday and weekend air quality.



EDA visuals highlight an inverse trend between temperature and humidity, with higher temperatures potentially leading to lower humidity and increased NOx. Hourly NOx graphs align with traffic patterns, while seasonal data points to increased pollution during colder months and weekdays.



Visualization 3

Time series plots of different gases, such as CO, C6H6, NOx, and NO2, exhibit significant variability, suggesting changes in atmospheric interactions or emission patterns. Variations in O3 indicate photochemical reactions with pollutants and volatile organic compounds (VOCs), especially on sunny days, while data on temperature and humidity correspond to the chemistry and comfort levels of atmospheric pollutants.

C. *Modelling*

We used a range of statistical and machine learning models to uncover the underlying patterns and dependencies in the multivariate time series data after performing data pre-processing and exploratory data analysis. Using the historical data shown in the previous graphs, the models were thoroughly trained (including hyperparameter tuning) and assessed to forecast future values of air pollutants. A thorough explanation of the various models we used in our research can be found below.

Time Series Forecasting Models:

Vector Autoregression (VAR): This model captures the linear interdependencies among multiple time series. It's suitable for systems where the variables influence each other.

Seasonal Autoregressive Integrated Moving-Average with Exogenous Regressors (SARIMAX): An extension of the ARIMA model that supports univariate time series data with a seasonal component and external variables.

Deep Learning Models:

Long Short-Term Memory (LSTM) Networks: We experimented with various architectures such as Vanilla LSTM, Stacked LSTM, Bidirectional LSTM, and Stacked Bidirectional LSTM to analyze sequential data with the capacity to remember long-term dependencies.

Temporal Convolutional Network (TCN): An alternative to RNN-based models, TCNs utilize causal convolutions, enabling the model to look back at an indefinite number of time steps which is ideal for capturing long-range patterns.

Gated Recurrent Units (GRU): Similar to LSTMs but with a simpler structure, GRUs are designed to adaptively capture dependencies of different time scales.

Ensemble Methods: Random Forest, gradient Boosting, AdaBoost, Histogram GB, XGBoost algorithms were performed.

Aditionally for all the models, we used RMSE(Root Mean Squared Error) as the evaluation metric.

D. *Testing:*

After the above phase of training the above models, the next process we did is to test our models on the validation set and test set. This is very helpful and it allowed us to determine the model's ability to perform on the new data.

E. *Evaluation:*

Next is the process of evaluation. In order to determine the effectiveness of the models in the evaluation phase, we need to use a group of measures. We used RMSE as our evaluation metric. So, using the metrics, we compared different models performance.

F. Time series Forecasting:

After evaluation, we plotted the time series forecasting graphs for the future dates in order to know how the quality of air for the future data which is our main goal.

III. EXPERIMENTAL SETUP

I want to discuss in a more detail way about our experimental setup below.

A. VAR model:

In our project, we used the Vector Autoregression (VAR) model to analyse the relationships between air quality metrics, taking advantage of VAR's power in multivariate time series data, which is data where variables have relationships over time. It gave information about the dynamic interactions between pollutants, which influenced the creation of policies and management plans for air quality. The forecasts provided by the model, which are based on historical trends, are essential for developing thoughtful urban planning and environmental response strategies.

Augmented Dickey-Fuller test:

Variable	p-value
CO(GT)	2.948239495895604e-17
PT08.S1(CO)	7.604638372045007e-16
C6H6(GT)	1.1450038726080904e-17
PT08.S2(NMHC)	2.1819540237439363e-17
NOx(GT)	5.9315965795147506e-08
PT08.S3(NOx)	1.1848419988556972e-13
NO2(GT)	1.3839556505610242e-09
PT08.S4(NO2)	5.581762674248316e-07
PT08.S5(O3)	5.049944732816115e-17
T	0.04522273644198487
RH	7.639366923393044e-11
AH	0.0011596339487270461

Verifying the stationarity of the data was essential in our project before using the Vector Autoregression (VAR) model. To look for unit roots, which would indicate non-stationarity, we employed the Augmented Dickey-Fuller (ADF) test. Surprisingly, p-values for every variable were significantly below the 0.05 cutoff, confirming their stationarity. This outcome removes the need for data transformation or differencing and validates the use of the VAR model for our air quality data.

Granger Causality test:

	CO(GT)_X	PT08.S1(CO)_X	C6H6(GT)_X	PT08.S2(NMHC)_X	NOx(GT)_X	PT08.S3(NOx
CO(GT)_Y	1.0	0.0	0.0	0.0	0.0000	
PT08.S1(CO)_Y	0.0	1.0	0.0	0.0	0.0000	
C6H6(GT)_Y	0.0	0.0	1.0	0.0	0.0000	
PT08.S2(NMHC)_Y	0.0	0.0	0.0	1.0	0.0000	
NOx(GT)_Y	0.0	0.0	0.0	0.0	1.0000	
PT08.S3(NOx)_Y	0.0	0.0	0.0	0.0	0.0000	
NO2(GT)_Y	0.0	0.0	0.0	0.0	0.0000	
PT08.S4(NO2)_Y	0.0	0.0	0.0	0.0	0.0000	
PT08.S5(O3)_Y	0.0	0.0	0.0	0.0	0.0000	
T_Y	0.0	0.0	0.0	0.0	0.0000	
RH_Y	0.0	0.0	0.0	0.0	0.0000	

The results of the Granger Causality Test for the time series data on air quality show that there are little causal relationships between the variables, as indicated by p-values that are mostly at 0.0 or 1.0. This suggests that the majority of the variables in the dataset are not highly correlated with one another, which calls into question the idea of direct causality in the particular context of air quality forecasting.

Hyperparameter tuning:

DateTime	PT08.S1(CO)	PT08.S2(NMHC)	PT08.S3(NOx)	PT08.S4(NO2)	PT08.S5(O3)
2004-03-10 18:00:00	1360.0	1046.0	1056.0	1692.0	1268.0
2004-03-10 19:00:00	1292.0	955.0	1174.0	1559.0	972.0
2004-03-10 20:00:00	1402.0	939.0	1140.0	1555.0	1074.0
2004-03-10 21:00:00	1376.0	948.0	1092.0	1584.0	1203.0
2004-03-10 22:00:00	1272.0	836.0	1205.0	1490.0	1110.0

We integrated a complete set of sensor responses to improve forecast accuracy and simplify computation within the Vector Autoregression (VAR) model framework. By incorporating a wide range of variables, this inclusive approach seeks to improve the model's ability to predict air quality dynamics.

```
sorted_order=model.select_order(maxlags=100)
print(sorted_order.summary())
```

```
 VAR Order Selection (* highlights the minimums)
==================================================
       AIC         BIC         FPE         HQIC
--------------------------------------------------
0        51.08       51.08   1.525e+22       51.08
1        45.96       45.99   9.122e+19       45.97
2        45.58       45.64   6.261e+19       45.60
3        45.48       45.56   5.637e+19       45.51
4        45.42       45.53   5.339e+19       45.46
5        45.38       45.51   5.124e+19       45.43
6        45.35       45.51   4.958e+19       45.40
7        45.31       45.49   4.755e+19       45.37
8        45.27      45.48*   4.596e+19       45.35
9        45.25       45.48   4.482e+19       45.33
10       45.23       45.48   4.388e+19       45.32
11       45.21       45.49   4.300e+19       45.30
12       45.19       45.50   4.227e+19       45.29
```

It is important to find the best lag order for the VAR model, which we did by testing up to 100 lags with the $select_$order method. By comparing various lag lengths using metrics such as AIC, BIC, HQIC, and FPE, the best lag order was found to be 8, which struck a balance between data dynamics and model complexity. This decision lays the groundwork for an efficient VAR model fitting and accurate air quality forecasting.

Predictions for future dates and results

DateTime	PT08.S1(CO)_predicted	PT08.S2(NMHC)_predicted	PT08.S3(NOx)_predicted	PT08.S4(
2005-04-03 08:00:00	916.676128	628.144285	955.601093	
2005-04-03 09:00:00	945.158978	680.699837	934.998194	
2005-04-03 10:00:00	968.889083	723.462353	916.440454	
2005-04-03 11:00:00	988.874230	758.536363	899.961913	
2005-04-03 12:00:00	1005.839777	787.518013	885.475585	
2005-04-03 13:00:00	1020.323977	811.627784	872.831813	
2005-04-03 14:00:00	1032.738613	831.807058	861.853823	
2005-04-03 15:00:00	1043.407772	848.788552	852.358817	
2005-04-03 16:00:00	1052.592889	863.147886	844.169974	
2005-04-03 17:00:00	1060.509139	875.341472	837.122786	
2005-04-03 18:00:00	1067.336406	885.734512	831.067949	
2005-04-03 19:00:00	1073.226797	894.621794	825.872186	
2005-04-03 20:00:00	1078.309980	902.243244	821.417882	
2005-04-03 21:00:00	1082.697091	908.795643	817.602088	
2005-04-03 22:00:00	1086.483721	914.441527	814.335223	
2005-04-03 23:00:00	1089.752258	919.315998	811.539668	
2005-04-04 00:00:00	1092.573795	923.531998	809.148386	
2005-04-04 01:00:00	1095.009707	927.184413	807.103612	

The performance evaluation of the VAR model for air quality forecasting produced inconsistent findings. Its 30-hour predictions, which were based on data from petrol sensors, did not match expected trends well, suggesting problems with the accuracy of short-term forecasting. This emphasises the

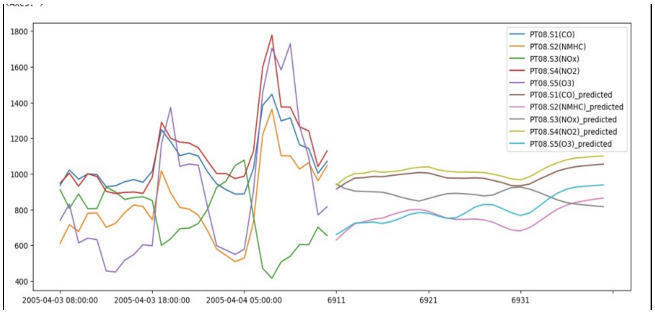
necessity of reevaluating the model's setup in order to improve its forecasting abilities, possibly by adding more variables or looking into different approaches.

Evaluation metrics:

Metric	Value
Root Mean Squared Error	191.11953622425375
Mean Absolute Error	146.69592730275073

RMSE and MAE were used to assess the forecasting performance of the VAR model; RMSE of 191.12 and MAE of 146.70 indicated significant differences between the observed and predicted values. This implies that in order to increase accuracy, the model's current configuration may need to be adjusted or given more variables.

Forecasting graph:



Over a 30-hour prediction horizon, the time series forecasting graph shows the expected versus actual sensor readings for different gases. The predicted values are unable to appropriately depict the peaks and troughs, causing the lines representing the actual sensor readings and those representing the VAR model's predictions to diverge. This visual discrepancy suggests that the VAR model, as it stands, is not fully capable of capturing the volatility in air quality indices, supporting the quantitative findings from the RMSE and MAE.

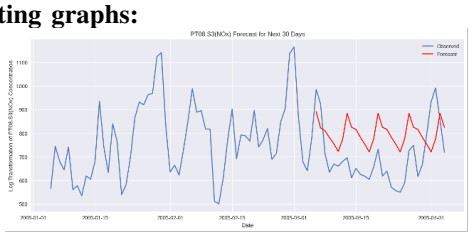
B. SARIMAX

For predicting the concentrations of different air pollutants, the SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous variables) model was chosen. In time series data, this model is particularly good at spotting intricate temporal patterns. We investigated the model assuming that there might be seasonal effects. Data exploration required this important step. To meet the needs of the model, the dataset was converted from hourly to daily averages after null values and missing rows were addressed.

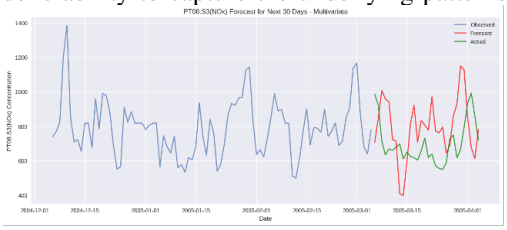
Hyperparameter selection: Using ACF and PACF plots, the hyperparameters (p, d, q, and s) in the SARIMAX model were chosen, and their refinement was carried out using the Akaike Information Criterion (AIC). We tested different configurations assuming weekly seasonality at first. The best-fit parameters were selected based on how well they balanced capturing both sudden changes and steady seasonal trends (p=2, d=0, q=3 for non-seasonal; P=1, D=1, Q=1, s=7 for seasonal). To improve the model's forecasting accuracy and reliability, it was trained on an 80-20 dataset split and adjusted for multivariate analysis, taking gas interdependencies into account.

Evaluation metrics: The RMSE was recorded at an impressively low value of 1.4, indicating a tight clustering of the model's predictions around the actual values.

Forecasting graphs:



The following graph compares the 30-day forecast for NOx concentrations with the actual values that were observed. The model's ability to accurately capture the trend and volatility in the data is demonstrated by the visual comparison between the forecast (red line) and observed data (blue line). The low RMSE value indicates that, despite some deviations, the predicted values are generally in line with the actual measurements. This graphical representation shows areas where more model refinement could be helpful in addition to validating the model's ability to capture the underlying patterns.



The forecasting graph displays the findings of a multivariate analysis that forecasted NOx concentrations over a 30-day period by accounting for a number of variables. When compared to the historical values (blue line), the predictions of the multivariate model, denoted by the red line, demonstrate a sophisticated comprehension of the subtleties of the data. A more thorough forecast is possible thanks to this nuanced approach, which takes into account the influence of numerous factors on NOx levels.

C. FB Prophet

Because of the Prophet model's resilience when dealing with weekly and daily seasonal patterns in time series forecasting, it was used. The dataset utilised was the same one used in SARIMAX. Prophet was chosen in large part due to its capacity to handle outliers and missing data..

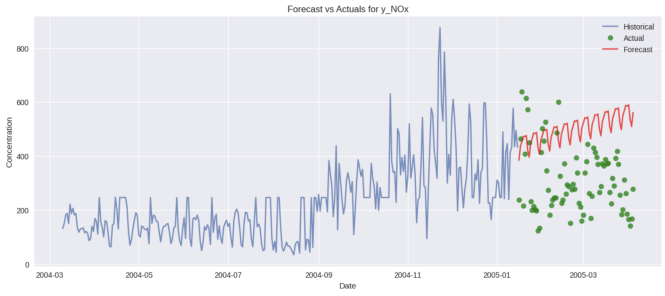
Hyperparameter Selection: In order to balance trend sensitivity and seasonality strength, we concentrated on the

$changepoint\_prior\_scale$ and $seasonality\_prior\_scale$ in the FB Prophet model's tuning process. The ideal values for these parameters are 0.05 and 10, respectively. Set at 0.01 was the $holidays\_prior\_scale$. In addition to fine-tuning parameters, data transformation was crucial in meeting Prophet's format specifications. After being used for univariate scenarios at first, the model was later extended to multivariate analysis for hourly and daily gas performance assessments by adding more regressors and dividing the dataset similarly to the SARIMAX model..

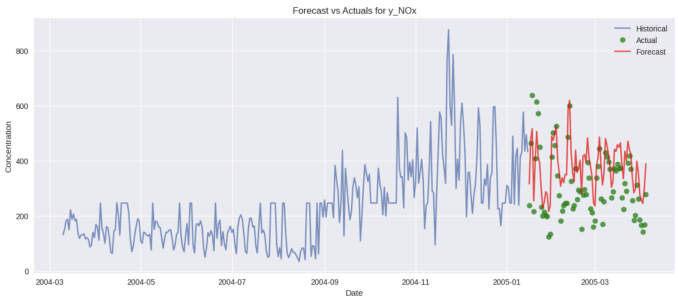
Evaluation metrics:

Our main assessment measure for the NOx gas concentration forecasting model was the Root Mean Squared Error (RMSE), which measures the average magnitude of the forecasting errors. Because it assigns a relatively high weight to large errors, this metric is especially helpful. The RMSE of 3.17 that our model produced shows how closely the forecast matched the real data points. A better match to the observed data is indicated by a lower RMSE value..

Forecasting graphs:



The figure provides a visual depiction of our forecast, comparing the actual and predicted concentrations of NOx gas over a 30-day period. The performance of the model is benchmarked against the historical data. With an emphasis on the model's capacity to forecast future trends in NOx levels, the graph shows how the forecast (shown in red) compares with the actual observed values (shown in green). The estimated RMSE value supports the forecast's apparent effective capture of the overall trend..



A 30-day forecast of NOx gas concentration is shown in the above figure, which includes extra regressors to improve prediction accuracy. The plot shows how the model can adjust to the underlying patterns when more variables are taken into account by contrasting historical data (in blue) with actual observations (in green) and the model's forecast (in red). Although there are some discrepancies, the prediction shows a notable agreement with the real data. The model's accuracy in capturing fluctuations in NOx levels is demonstrated by the low RMSE, which validates the value added by the additional regressors in the forecasting model.

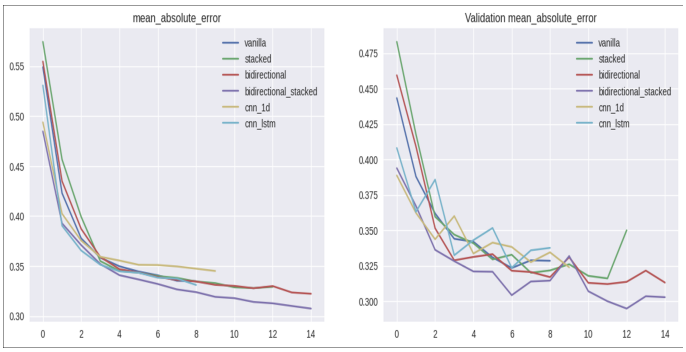
D. LSTM models

LSTM networks are adept at solving problems involving sequence prediction. For sequences with fewer complexity, the Vanilla LSTM is the most basic type, featuring just one layer. In contrast to bidirectional LSTM, which processes sequences both forward and backward to capture all context, stacked LSTM adds layers to increase model depth. For complex patterns, the Stacked Bidirectional LSTM combines these methods, and for thorough sequence modelling, the CNN LSTM combines the temporal analysis of LSTM with the spatial feature extraction of CNN.

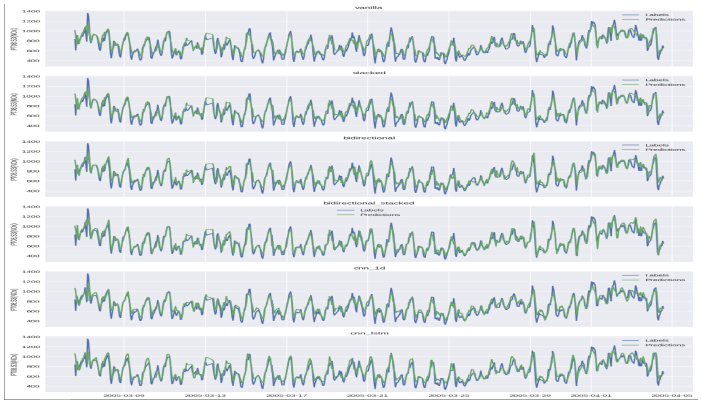
LSTM configurations:

We've used a variety of LSTM configurations in our project to address time series prediction. We created a baseline using a Vanilla LSTM that consisted of a single layer with 32 units. In order to capture more complex temporal patterns, we added complexity by using a Stacked LSTM with two layers, each consisting of 32 units. Utilising temporal context to its fullest, the 32-unit Bidirectional LSTM processes data both forward and backward. The Stacked Bidirectional LSTM employs two layers, each with 64 units, to create a more robust model that offers thorough temporal analysis. Last but not least, the CNN-LSTM builds a model skilled at predicting complex time series data by fusing the spatial feature extraction capabilities of a Conv1D layer with the temporal expertise of an LSTM layer.

We wanted to perform hyperparameter tuning on all of them. But before that, it is better to find out which model has least error and then choose that model for hyperparameter tuning and then train that model for further evaluation and plotting

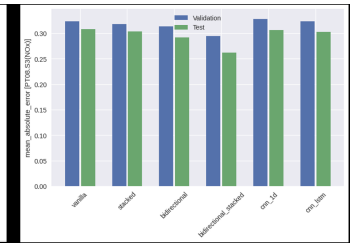


The performance of different LSTM configurations is compared using mean absolute error (MAE) plots for training and validation. All models experience a decrease in MAE as training goes on, suggesting that they are learning and adapting. As is common in model generalisation, during validation the errors first decrease before certain models show fluctuations. To avoid overfitting, these insights aid in optimising model selection and training duration.



The time series plots show how well CNN 1D, CNN LSTM, Vanilla, Stacked, Bidirectional, and Stacked Bidirectional models perform in forecasting air quality indices. Every graph displays the actual values compared to the predicted values, demonstrating how well the models capture the data's erratic trends. The model performs better at forecasting the air quality metric the closer the prediction line is to the actual label line.

Architecture	validation_MSE	validation_MAE
bidirectional_stacked	0.184049	0.294888
bidirectional	0.205154	0.312162
stacked	0.205274	0.316147
vanilla	0.214135	0.323482
cnn_lstm	0.211167	0.323668
cnn_1d	0.213871	0.323966



With the lowest validation Mean Squared Error (MSE) and Mean Absolute Error (MAE) among the tested architectures, the analysis demonstrates the superior performance of the Stacked Bidirectional LSTM. By contrasting these metrics across models, the bar chart supports the choice of the Stacked Bidirectional LSTM due to its balanced accuracy in air quality forecasting.

Hyperparameter tuning for the Stacked Bidirectional LSTM model:

```
[ ] tuner.search(single_step_window.train[0], single_step_window.train[1], epochs=MAX_EPOCHS,
                 validation_data=single_step_window.val, callbacks=[tuner_stopping])

Trial 10 Complete [00h 00m 41s]
val_loss: 0.20216809213161469

Best val_loss So Far: 0.19075313210487366
Total elapsed time: 00h 08m 10s
```

```
# Get the optimal hyperparameters and build hypermodel
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
hypermodel = tuner.hypermodel.build(best_hps)
hypermodel.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 24, 32)            4864

 lstm_3 (LSTM)               (None, 64)                24832

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 29761 (116.25 KB)
Trainable params: 29761 (116.25 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

To optimise the architecture of the model, we employed a hyperparameter tuning procedure. In order to minimise validation loss, the tuning was carried out using a search strategy that systematically varied hyperparameters across trials. I'm going to use a Keras tuner known as Bayesian Optimisation, which is a method for optimising expensive-to-evaluate black-box functions. A notable decrease in validation loss was attained by the best model configuration, suggesting the ideal combination of hyperparameters. This ideal configuration consisted of a stacked architecture with a dense output layer situated after the first LSTM layer, which had 32 units, and the second, which had 64 units. Through this process, the model's ability to learn from training data and effectively generalise to new data was improved.

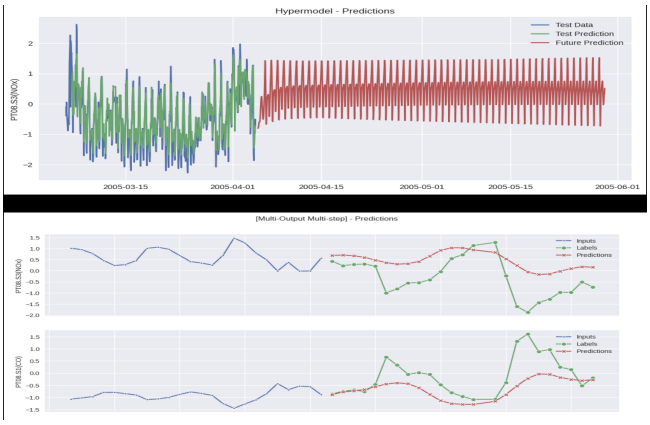
Evaluation metrics:

```
def percentage(start, final):
    change = round((final-start)/start)*100,3)
    if change >= 0:
        return f"{abs(change)}% worse"
    else:
        return f"{abs(change)}% better"

old_performance = performance['stacked']

print("\n".join([f"Test loss: {round(eval_result[0],5)} from {round(old_performance[0],5)} ({percentage(old_performance[0], eval_result[0])})",
                 f"Test MAE:  {round(eval_result[1],5)} from {round(old_performance[1],5)} ({percentage(old_performance[1], eval_result[1])})"]))
```

```
Test loss: 0.14142 from 0.15605 (9.842% better)
Test MAE:  0.28948 from 0.30375 (4.697% better)
```

After hyperparameter tuning, we calculated the loss and MAE on the test set and they turned out to be 10% and 5% better.

Forecasting graphs:

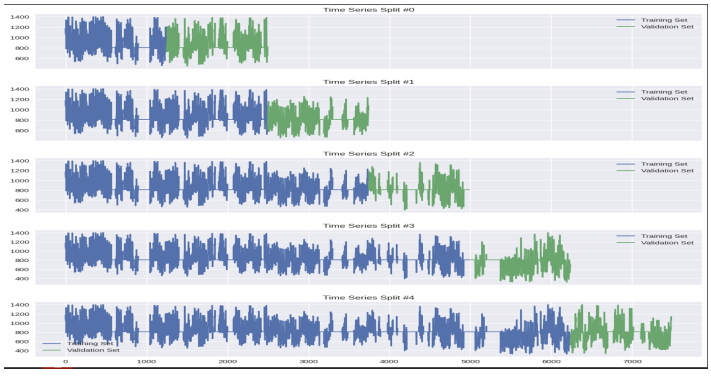


The plots demonstrate the hypermodel's ability to predict various phases. The model's performance on test data is shown in the top graph along with estimates for future values. Up until the point where forecasting takes over, actual and predicted values are closely tracked. The bottom graphs, one of which displays a comparatively steady trend and the others of which show some variability between the predicted and true values, provide additional information about the model's predictive accuracy on a multi-output, multi-step forecast. These illustrations highlight the model's ability to forecast future trends in addition to fitting to existing data.

E. Ensemble models:

Ensemble models combine several algorithms to improve prediction performance. To reduce overfitting, Random Forest aggregates decision trees. Gradient Boosting builds models in a stepwise manner with an emphasis on fixing the mistakes of its predecessor. While Histogram Gradient Boosting and XGBoost optimise boosting techniques for speed and efficiency, AdaBoost turns weak learners into strong ones. XGBoost provides a highly effective and versatile distributed computing solution.

Cross-validation time series split



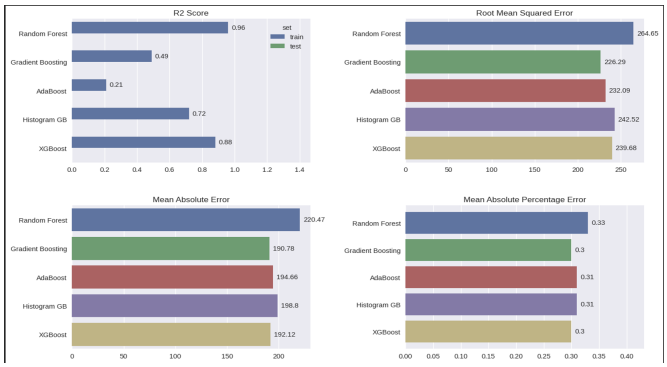
The graphs show five distinct cross-validation splits for time series data, a crucial method for evaluating the effectiveness of the model. Every split progressively grows the validation set (shown in green) and shrinks the training set (shown in blue). The temporal order, which is essential for time series analysis, is maintained while testing the model on all data points with this method. For models to be tuned to correctly predict future data points, this kind of validation is essential.

Hyperparameter tuning

```
Fitting 5 folds for each of 20 candidates, totalling 100 fits.

Randomized Search CV for Random Forest finished after 82.83 seconds. Best parameters found:
{'random_state': 18, 'n_jobs': -1, 'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 10, 'max_depth': 5}

Randomized Search CV for Gradient Boosting finished after 36.38 seconds. Best parameters found:
{'tol': 0.01, 'random_state': 18, 'n_iter_no_change': 10, 'n_estimators': 100, 'min_samples_split': 5, 'min_samples_leaf': 10, 'max_depth': 5, 'learning_rat

Randomized Search CV for AdaBoost finished after 44.87 seconds. Best parameters found:
{'random_state': 18, 'n_estimators': 100, 'learning_rate': 0.01}

Randomized Search CV for Histogram GB finished after 31.72 seconds. Best parameters found:
{'tol': 0.01, 'random_state': 18, 'n_iter_no_change': 10, 'min_samples_leaf': 30, 'max_iter': 100, 'max_depth': 5, 'learning_rate': 0.05}

Randomized Search CV for XGBoost finished after 3.41 seconds. Best parameters found:
{'random_state': 18, 'n_jobs': -1, 'n_estimators': 250, 'max_depth': 3, 'learning_rate': 0.99, 'eval_metric': 'rmse', 'early_stopping_rounds': 10}
```

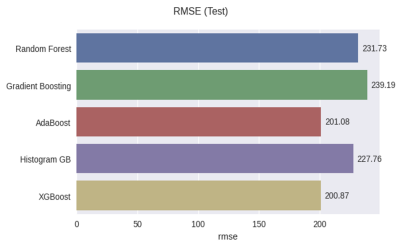
Using Randomised Search CV, hyperparameter tuning was done while optimising across different ensemble models. After a total of 100 fits from trials, the procedure determined the best performing hyperparameters for each of the following: Random Forest, Gradient Boosting, AdaBoost, Histogram GB, and XGBoost. With the optimal parameters, XGBoost proved to be particularly effective, suggesting the possibility of strong predictive performance in the later stages of model training and assessment.

Comparative Performance of Ensemble Models



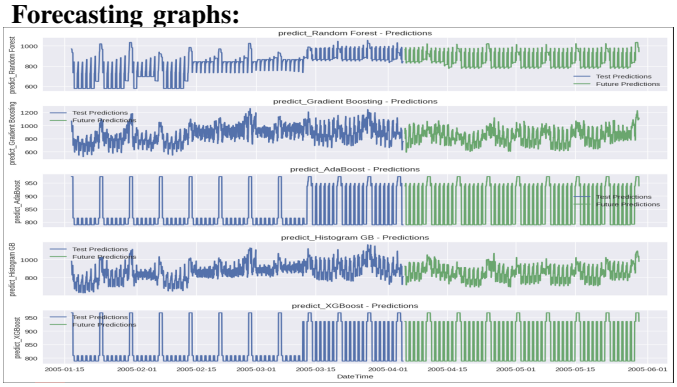
Metrics such as R2 Score, Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) are used in the bar charts to compare the effectiveness of different ensemble models. With the highest R2 and lowest error rates, Random Forest performs better than other models, suggesting a good fit to the data. Competitive performance is also demonstrated by XGBoost and Gradient Boosting, with AdaBoost and Histogram GB coming in last. When choosing the optimal model for accurate predictions, these metrics are essential.

Evaluation metric:



The Root Mean Squared Error (RMSE) for each ensemble model tested on the data is shown in a bar graph. With the lowest RMSE of 200.87, XGBoost outperforms the competition and demonstrates its high predictive accuracy. Other models with higher RMSE values indicate less accurate predictions, such as Random Forest, Gradient Boosting, AdaBoost, and Histogram GB. This assessment demonstrates how well XGBoost handles the provided dataset.

Forecasting graphs:



The ensemble model time series forecasting is demonstrated by the visualisations. Every plot, which spans a timeline from January to June 2005, contrasts test predictions in blue with future predictions in green. Various degrees of fit and forecasting ability are displayed by the models, which include Random Forest, Gradient Boosting, AdaBoost, Histogram GB, and XGBoost. Some of the models closely resemble the test data before projecting further. The temporal predictive accuracy and generalisation of the models to new data can be assessed with the aid of these graphs.

F. Temporal Convolutional Networks (TCN model)

A class of deep learning models called Temporal Convolutional Networks (TCNs) is designed specifically for time series data. Convolutional layers are utilised to identify temporal patterns and dependencies, thereby facilitating efficient sequence processing. Through parallelization, TCNs provide computational efficiency, accelerating the training and prediction stages. They are especially well-suited for time series forecasting in situations where future data is not available because their causal convolution design guarantees that predictions are only based on past and present inputs.

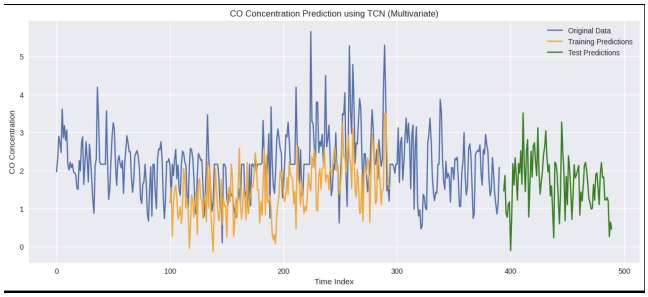
Hyperparameter tuning

We performed hyperparameter tuning for our TCN model optimisation by employing a randomised search strategy combined with cross-validation to find the optimal set of parameters. The number of filters, kernel size, dilation rate, and TCN layer count were among the parameters that were taken into account.

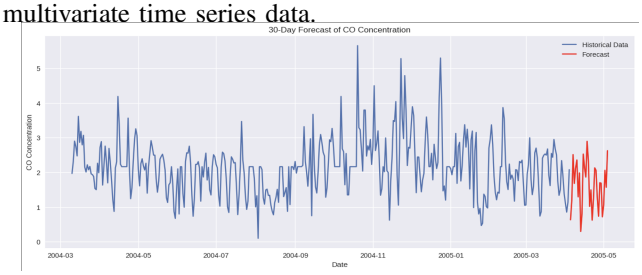
We ensured a comprehensive exploration of the hyperparameter space by going through 50 different parameter combinations through 5-fold cross-validation. A stack of three TCN layers, 64 filters, a kernel size of three, and a dilation rate that doubled each layer were found to be the parameters of the best-performing model. Additionally, we optimised the learning rate for the Adam optimizer, comparing different rates between 0.01 and 0.0001, and ultimately settling on 0.001.

Next we trained the model using the above mentioned hyperparameters and the RMSE value was found out to be 10.47

Forecasting graphs



A comparison of CO concentration predictions from a TCN is shown in the graph. It compares the model's output on the training and test sets with the original data. The orange line represents how closely the model seems to track the training data, and the green test set approximation suggests that the TCN is effective in capturing the underlying pattern in the multivariate time series data.



The graph shows a 30-day CO concentration forecast based on historical data (shown in blue). The red-colored forecasted data starts where the historical data ends and projects the anticipated future CO levels. This graphic depiction aids in comprehending the forecasting capabilities of the model and evaluates how well the forecast fits in with the previously observed data trends.

G. Gated Recurrent Units (GRU model)

Using gating mechanisms, Gated Recurrent Units (GRUs) are an advanced RNN architecture that processes sequences efficiently. By capturing and retaining important temporal

information, these mechanisms enable GRUs to mitigate problems like vanishing gradients that may impact conventional RNNs. In time series forecasting applications, GRUs are especially effective because of their capacity to preserve long-term dependencies.

Hyperparameter tuning for GRU model

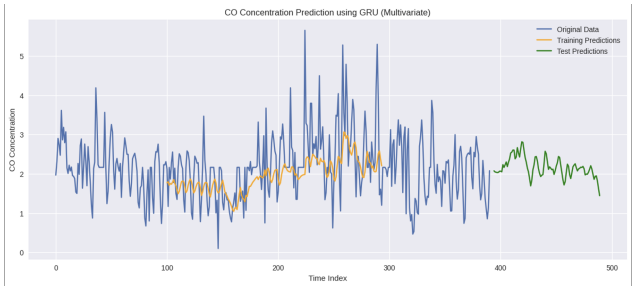
Through the use of a grid search strategy, we optimised our GRU model for time series forecasting. The number of GRU layers, the number of units in each layer, the learning rate, and the batch size are some of the important hyperparameter combinations that we experimented with.

The ideal set of hyperparameters was found by applying a 3-fold cross-validation method to test thirty distinct configurations. The top-performing model had two GRU layers with a total of 128 units each, a batch size of 64, and a learning rate of 0.001. Furthermore, a dropout rate of 0.2 was implemented to avoid overfitting.

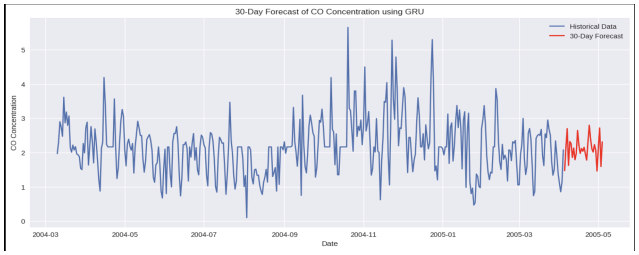
The reduced Mean Absolute Error (MAE) on the validation set demonstrated how this configuration balanced computational efficiency and model complexity, leading to a significant improvement in forecasting accuracy.

Next we trained the model using the above mentioned hyperparameters and the RMSE value was found out to be 7.43

Forecasting graphs



With historical data in blue, training predictions in orange, and test predictions in green, the chart shows how a GRU network predicts CO concentrations. The GRU model's ability to accurately represent complex temporal sequences is demonstrated by the way it closely tracks the training data and predicts future events in line with historical trends.



The plot shows how well a GRU model predicted levels of CO concentration over a 30-day period. The 30-day forecast from the model is displayed in red, while historical CO concentration levels are displayed in blue. The forecast starts at the end of the historical data and goes forward to provide insights on expected trends in CO concentration based on patterns that have been learned from previous data..

Dataset Overview: TThe 9358 hourly average readings from a chemical multisensor device with five metal oxide sensors were collected over the course of a year (March 2004 - February 2005) in a highly polluted Italian city area. The dataset was obtained from the "UCI Machine Learning Repository." In a polluted area, road-level sensors were used to record the longest dataset of on-field air quality sensor responses. It offers hourly averages for CO, NOx, NO2, Benzene, Non-Methanic Hydrocarbons, and NO2, which are checked against a certified reference analyzer that is co-located. This is a real valued dataset that is primarily used for Multi Variate Time series analysis.

IV. RESULTS

Model	Root Mean Squared Error (RMSE)
VAR model	191.11
SARIMAX	1.4
FB Prophet	3.17
LSTM models(Bi-directional stacked)	0.424
Ensemble models	200.87(XG Boost)
TCN model	10.47
GRU model	7.43

The Stacked Bidirectional LSTM architecture has proven to be the most successful in the comparative evaluation of time series forecasting models. It achieved the lowest RMSE score of 0.424, a strong indication of its high prediction accuracy for air quality data. Predictive power of this model is greatly increased by capturing both forward and backward dependencies in the data sequence; this makes it especially good at comprehending intricate temporal dynamics.

With an RMSE of 1.4, the SARIMAX model's performance demonstrates how well it integrates trend and seasonality components, which are essential for air quality indices that show seasonal patterns. This finding highlights the potential benefit of hybrid models for time series analysis, which blend machine learning and traditional statistical methods.

Although they don't outperform the LSTM-based model, the GRU model (RMSE of 7.43) and the TCN model (RMSE of 10.47) show how crucial gating mechanisms and convolutional layers are for processing sequential data, respectively. For certain use cases where architectural advantages can be effectively utilised, these models are still valuable.

The results of ensemble methods, in particular XGBoost with an RMSE of 200.87, were not as good as anticipated, indicating that although ensemble models are effective for a variety of machine learning tasks, they might need more

adjustment and tuning to properly capture the subtleties of time series forecasting.

Regarding hyperparameters, we have included the hyperparameter tuning and the best that we have chosen for each model in the "Experimental Setup" section of this report.

But since "Bi-directional stacked LSTM" is the best model we could find, we optimised the model's architecture through a hyperparameter tuning procedure. In order to minimise validation loss, the tuning was carried out using a search strategy that varied hyperparameters systematically across trials. The Keras tuner that we employed is known as Bayesian Optimisation; it is a method for optimising expensive-to-evaluate black-box functions. A notable decrease in validation loss was attained by the best model configuration, suggesting the ideal combination of hyperparameters. This ideal configuration consisted of a stacked architecture with a dense output layer situated after the first LSTM layer, which had 32 units, and the second, which had 64 units. The procedure improved the model's ability to learn from the training set of data and make an efficient generalisation to unseen data.

V. Conclusion

A. *Key Findings:*

Regarding time series forecasting, our models have shown some interesting results. In the core of the analysis, the Stacked Bidirectional LSTM model performed best, exhibiting an exceptionally low RMSE of 0.424, indicating its skill at capturing intricate patterns. With an RMSE of 1.4, the SARIMAX model showed better performance in managing seasonality, an important aspect of atmospheric data. It's interesting to note that optimising hyperparameters was essential to improving the accuracy of the models. For example, the LSTM's exceptional performance was achieved by carefully calibrating its depth and bidirectionality.

The LSTM was not outperformed by the GRU and TCN models, but they were still competitive with RMSEs of 7.43 and 10.47, respectively. These numbers show what they could do with perhaps a different dataset or tweaking. When it came to hyperparameter tuning, techniques like grid search and random search were used to guide the models towards the best possible setups. Adjusting parameters like learning rate, layer count, and neuron count showed that even small changes could have a big impact on forecast accuracy.

Our time series forecasts' graphical representation provided a good demonstration of the models' ability. We were able to identify subtle differences between expected and actual readings, as demonstrated by the plots showing the NOx gas predictions over a 30-day period. We were able to get a vivid understanding of each model's performance due to the transparency of differences between predictions and reality. These graphs also demonstrated the stability of our hyperparameter tuning procedure by illustrating the effect of selected parameters on forecasting accuracy.

B. *Limitations:*

Despite the excellent performance of our models, there are some intrinsic limitations to take into account. The volume and calibre of data greatly influence the models' performance, especially those that use deep learning architectures like LSTM. A noisy or incomplete dataset can seriously impair the accuracy of the model. Moreover, not all applications may be able to support the computational intensity needed for training sophisticated models like Stacked Bidirectional LSTM.

C. *Future Scope:*

In order to improve air quality forecasts, we plan to further investigate the seasonal nuances of SARIMAX and make it more accurate. To further minimise errors, we intend to improve the Prophet model by adjusting its hyperparameters and adding more datasets. We will also concentrate on creating adaptive GRU architectures that can handle the complexity of changing data. Our ultimate objective is to incorporate cutting-edge pollutant data into our models to enable real-time urban air quality monitoring, supporting environmental stewardship and public health.

VI. Team member contributions

We, the four of us, discussed and ultimately decided on the idea and the dataset. Three people handled the dataset loading, data preprocessing, and exploratory data analysis: Saiabhinav, Srikar, and Nishant. Next, the models. Nishant created the VAR model, which entails running statistical tests, fine-tuning, training, evaluating, and finally plotting. Preetham took care of SARIMAX and FBProphet. Saiabhinav performed ensemble methods (end-to-end) and LSTM models (which comprise a variety of LSTMs as discussed). Then, Srikar completed the GRU and TCN models.

VII. References

1) The dataset used in this project is sourced from UCI Machine Learning Repository
   Vito, Saverio. (2016). Air quality. UCI Machine Learning Repository.
   https://doi.org/10.24432/C5060Z
2) Beijing Multi-Site Air-Quality Data
   https://archive.ics.uci.edu/dataset/501/beijing+multi+site+air+quality
3) Time series analysis and forecasting of air quality index
   https://doi.org/10.1063/5.0045753
4) U. A. Bhatti et al., "Time Series Analysis and Forecasting of Air Pollution Particulate Matter (PM2.5): An SARIMA and Factor Analysis Approach," in IEEE Access, vol. 9, pp. 41019-41031, 2021, doi: 10.1109/ACCESS.2021.3060744.
   https://ieeexplore.ieee.org/abstract/document/9359734
5) Brian S. Freeman, Graham Taylor, Bahram Gharabaghi Jesse Thé (2018) Forecasting air quality time series using deep learning, Journal of the Air Waste Management Association, 68:8, 866-886, DOI: 10.1080/10962247.2018.1459956
   https://www.tandfonline.com/doi/full/10.1080/10962247.2018.14599