A COMPARATIVE ANALYSIS OF FIVE TIME SERIES MODELS FOR CO_2 EMISSIONS FORECASTING IN PORT-HARCOURT AND ITS ENVIRONS

Engr. Orimadike Okechukwu Stanislaus
Centre for Information and Telecommunications Engineering
University of Port Harcourt, **NIGERIA**
orimadikeokechukwu@gmail.com

Dr. Okengwu Ugochi Adaku
HOD, Computer Science
University of Port Harcourt
NIGERIA
ugochi.okengwu@uniport.edu.ng

Prof. Omijeh Bourdillon Odianonsen
Director, CITE, University of Port Harcourt
NIGERIA
bourdillon.omijeh@uniport.edu.ng

ABSTRACT

Accurate forecasts of greenhouse gas (GHG) emissions are crucial for addressing climate change and guiding effective mitigation strategies. We developed and tested advanced techniques to improve time-series GHG emissions forecasting, addressing the limitations of existing models. Our study explored various algorithms, including ARIMA, SARIMA, ETS, Prophet, and TBATS, to identify the most effective methods for capturing the complex seasonality and non-linear patterns in GHG data particular to the city of Port Harcourt, Rivers State Nigeria. We tested the stationarity of the time series using ADF and KPSS tests. The ETS model, selected for its ability to handle trend and seasonal components, was optimized using grid search and the Akaike Information Criterion (AIC). We then compared its performance with ARIMA, SARIMA, Prophet, and TBATS models. The ETS model outperformed the others, with predicted CO_2 values within the 95% confidence interval of observed data and a mean absolute error (MAE) of 14.82 and a root-mean-square error (MSE) of 18.91. This research marks a significant advancement in GHG emissions forecasting, underscoring the practical value of well-tuned models in environmental science and their relevance to policy decisions. Future work should focus on refining these models for real-time use, ensuring a balance between computational efficiency and predictive accuracy to provide actionable insights for policymakers and environmental scientists.

Keywords: Time Series, Emission, Greenhouse gases, Forecasting, Exponential Smoothing.

INTRODUCTION

Anticipating greenhouse gas (GHG) emissions is essential for devising effective strategies to mitigate climate change. In 2023, the International Energy Agency (IEA) reported a 1.1% increase in global energy-related CO_2 emissions, reaching a record high of 37.4 gigatonnes (Gt) (IEA, 2023). This underscores the importance for climate mitigation authorities to deeply understand current and future emissions trends in order to develop and implement effective countermeasures. This paper reviews predictive models and algorithms used for GHG emissions forecasting, examining their strengths and weaknesses as highlighted in previous studies. The focus is on understanding how research into these inefficiencies can guide improvements in forecasting accuracy.

Greenhouse gases (GHGs) are atmospheric gases that trap and re-emit infrared radiation, causing the greenhouse effect. While this effect is crucial for maintaining life-sustaining temperatures on Earth, human activities such as burning fossil fuels and deforestation have significantly increased GHG emissions, leading to global warming and climate change (British Geological Survey, 2023). Carbon dioxide (CO_2), the primary greenhouse gas, is largely produced by burning fossil fuels like coal, oil, and natural gas, as well as through industrial processes (Nutongkaew et al., 2014). Methane (CH_4), another potent GHG, primarily originates from coal, oil, and natural gas production, as well as from agricultural activities such as livestock digestion and manure decomposition (Smith et al, 2021). Nitrous oxide (N_2O) emissions come from sources like fossil fuel combustion, agricultural processes, and waste treatment (Overview of Greenhouse Gases | US EPA, 2024). As these gases accumulate in the atmosphere, they intensify the greenhouse effect, trapping more heat and causing global temperatures to rise, which in turn leads to environmental consequences such as rising sea levels, shifting weather patterns, and more frequent and severe storms.

Forecasting future trends is a crucial aspect of environmental science, as it allows for the extrapolation of current data to predict future outcomes. Forecasting involves analyzing past and present data using methods like time series analysis, regression analysis, or machine learning algorithms (Bodily et al., 2024). Accurate forecasts are vital for making informed decisions across various fields, including business, weather prediction, supply chain management, and environmental planning. In environmental science, time series analysis is particularly valuable for examining data trends over time, such as GHG emissions. This type of analysis uses statistical methods to identify patterns like trends and seasonality within a sequence of data points recorded over time. Common models used for time series forecasting include Autoregressive Integrated Moving Average (ARIMA), Exponential Smoothing, and Seasonal Decomposition of Time Series (STL) (Adhikari et al, 2013). These models are essential tools for predicting future GHG emissions and other environmental variables.

The aim of our research is to explore various algorithms, including ARIMA, SARIMA, ETS, Prophet, and TBATS, to identify the most effective methods for capturing the complex seasonality and non-linear patterns in GHG data.

Seasonality, a key feature of time series data, refers to regular, periodic fluctuations that occur at specific intervals, such as weekly, monthly, or quarterly. Recognizing seasonality is crucial for accurate forecasting, as it can significantly impact the interpretation of data and the accuracy of predictions. For instance, businesses often experience predictable seasonal variations in sales, which must be accounted for when forecasting future performance.

LITERATURE REVIEW

The aim of our research is to address the limitations and gaps in current forecasting models for greenhouse gas (GHG) emissions, as highlighted in recent studies. Zhang et al. (2019) utilized ARIMA and SARIMA models to forecast CO_2 emissions in China, finding that SARIMA provided more accurate forecasts by better capturing seasonality. However, these models struggled with non-linear patterns and sudden changes in emission trends. Wang et al. (2020) applied an Artificial Neural Network (ANN) to forecast GHG emissions in the U.S., effectively modeling complex non-linear relationships for long-term forecasting. However, this approach required a large training dataset and significant computational resources.

Li et al. (2021) used Support Vector Machines (SVM) for predicting CO_2 emissions in India. The SVM model demonstrated good accuracy and robustness, particularly in handling high-dimensional datasets, making it well-suited for regions with heterogeneous emission sources. Pao et al. (2018) developed a hybrid ARIMA-ANN model for GHG emissions prediction in Taiwan, achieving higher accuracy by simultaneously detecting linear and non-linear patterns in the data. Similarly, Chen et al. (2020) combined wavelet transforms with ANN to model CO_2 emissions in South Korea, effectively capturing short-term fluctuations and long-term trends.

In the United Kingdom, Taylor and Letham (2018) employed the Prophet model to predict CO_2 emissions, finding it effective even in the presence of missing data and seasonal variations. De Livera et al. (2011) utilized the TBATS model to forecast GHG emissions in Australia, successfully capturing complex cyclical patterns for long-term, high-quality forecasts.

Despite these advancements, several gaps remain in GHG emissions forecasting. First, there is a limited integration of multiple data sources, with existing models typically relying on single data sources with restricted sample sizes or geographical scopes. This can lead to misleading conclusions. Integrating diverse data sources, such as historical emissions, economic indicators, and environmental variables, could enhance the robustness of these models. Additionally, there are challenges in comparing various models, as many studies focus on a single model without methodically evaluating multiple options to determine the best fit for the data and domain.

Scalability and computational efficiency are also significant concerns, as some machine learning models are too computationally expensive for real-time applications. Research should focus on optimizing these models to ensure they can scale without losing accuracy. Finally, while models like SARIMA address seasonality, they often struggle with more subtle, nuanced patterns. Advanced models such as the Exponential Smoothing State Space Model (ETS), Prophet, and TBATS offer better handling of seasonality and should be considered for future research.

Our research explores the under-listed forecasting models and algorithms used for predicting greenhouse gas (GHG) emissions, highlighting their strengths, limitations, and suitability for different types of data.

The Autoregressive Integrated Moving Average (ARIMA) model is a commonly used method that combines autoregression, differencing, and moving averages. It is effective for short-term forecasts, particularly for time series data with linear trends. The ARIMA model can be mathematically expressed as:

$$(1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p)(1 - B)^d y_t = (1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q)\varepsilon_t$$

Where:

- $y_t$is the current value of the series.
- ϕ_i for $i = 1, 2, \cdots, p)$are the autoregressive parameters.
- θ_i for $i = 1, 2, \cdots, p)$ are the moving average parameters.
- B is the lag operator.
- ε_t is the white noise error term.

However, ARIMA struggles with capturing seasonal variations and non-linear patterns, which are often present in GHG emission data.
To address ARIMA's limitations in dealing with seasonality, the Seasonal ARIMA (SARIMA) model was developed. SARIMA extends ARIMA by incorporating seasonal differencing and additional parameters to model seasonal components of the time series. The SARIMA model is expressed as:

$$\phi_p(B)\Phi_P(B^s)(1 - B)^d(1 - B^m)^D y_t = \theta_q(B)\Theta_Q(B^m)\varepsilon_t$$

Where:

- Φ_P and $\Theta_Q$are seasonal autoregressive and moving average coefficients, respectively.

- m represents the number of time steps in a seasonal period.

This model is more suitable for forecasting GHG emissions, which often exhibit seasonal fluctuations. Despite this improvement, SARIMA still faces challenges in capturing non-linear trends and sudden changes in emission patterns.
The Exponential Smoothing State Space Model (ETS) provides a flexible approach to forecasting by considering error, trend, and seasonality components. The ETS model is represented by:

$$y_t = l_{t-1} + b_{t-1} + \varepsilon_t$$
$$l_t = l_{t-1} + b_{t-1} + \alpha\varepsilon_t$$
$$b_t = b_{t-1} + \beta\varepsilon_t$$

Where:
- l_t and $b_t$are the level and trend components.
- α and β are smoothing parameters.

ETS adapts to changes in seasonal patterns over time, making it useful for forecasting GHG emissions with varying seasonal characteristics.
For more complex seasonal patterns and non-linearities in time series data, the Trigonometric Box-Cox ARMA Trend Seasonal (TBATS) model is particularly effective. The TBATS model uses trigonometric functions to capture seasonality and a state-space approach for parameter estimation. The model is represented as:

$$y_t(\lambda) = l_t + \phi b_t + \sum_{k=1}^{m} \gamma_{t,k} + d_t + \epsilon_t$$

Where:
- $\gamma_{t,k}$ are the trigonometric seasonal components.
- -d_t is the ARMA error term.

This makes TBATS a promising framework for GHG emissions forecasting, especially when dealing with complex seasonal patterns.
Prophet, developed by Facebook, is a user-friendly forecasting tool designed for non-experts. It models non-periodic changes in time series data using a piecewise linear or logistic growth curve, while also accounting for seasonal variations and holiday effects. The Prophet model is expressed as:

$$y(t) = g(t) + s(t) + \varepsilon_t$$

Where:
- g(t) represents non-periodic changes.
- s(t) captures periodic changes.

In summary, each of these models—ARIMA, SARIMA, ETS, TBATS, and Prophet—offers unique strengths and weaknesses for forecasting GHG emissions. ARIMA and SARIMA work well for simpler, linear time series with or without seasonality, while ETS offers flexibility for changing seasonal patterns. TBATS is ideal for handling complex seasonalities and non-linearities, making it suitable for more intricate datasets. Prophet balances flexibility and ease of use, catering to diverse seasonal and trend variations. Collectively, these models provide a comprehensive toolkit for addressing the challenges in forecasting GHG emissions, with each model offering specific advantages based on the data and forecasting needs.
.

METHODOLOGY

The above-discussed models were developed and implemented using Python in Google Colab, leveraging various statistical libraries for their construction and evaluation. Google Colab facilitated the creation of a cloud-based research platform. The libraries utilized include Numpy, Python, Scikit-learn, and Statsmodels. Figures 1 and 2 illustrate the framework and machine learning process as follows:

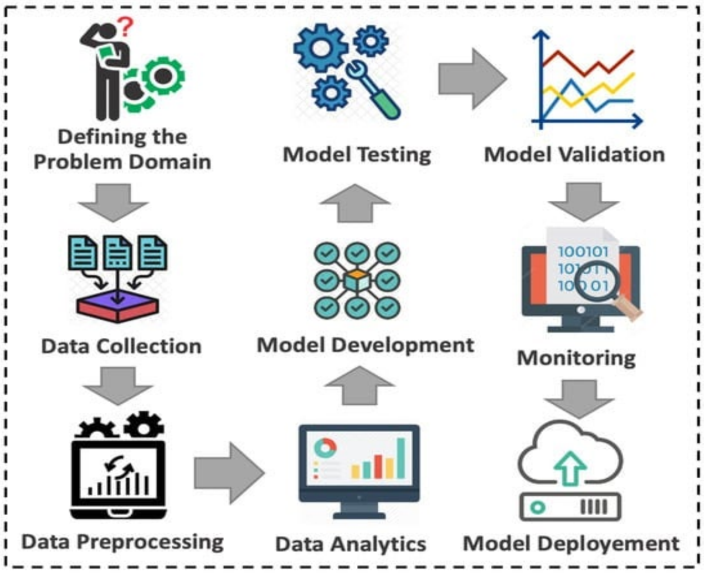


Figure 1: Machine learning framework. (Surakhi, O et al., 2021)

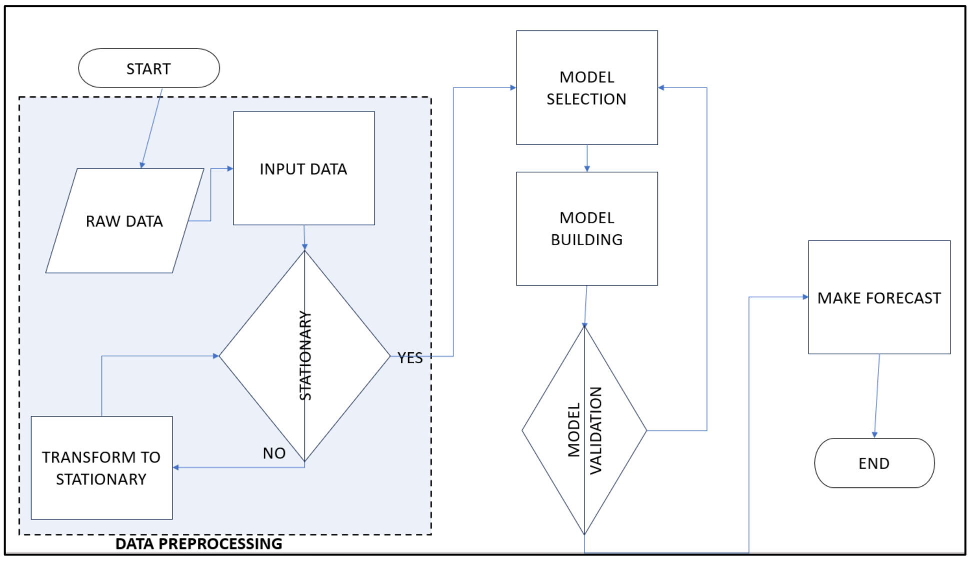


Figure 2: Flow Diagram for Model Building and Evaluation

Problem Domain

We defined our problem domain as evaluating the effectiveness of the specified time series models for the chosen location of Port Harcourt, Rivers State, Nigeria.

Data Collection

- The next step was to initiate data collection.
- Data was gathered continuously over several months to capture both short-term fluctuations and long-term trends in greenhouse gas emissions.

Dataset Composition

The dataset includes:

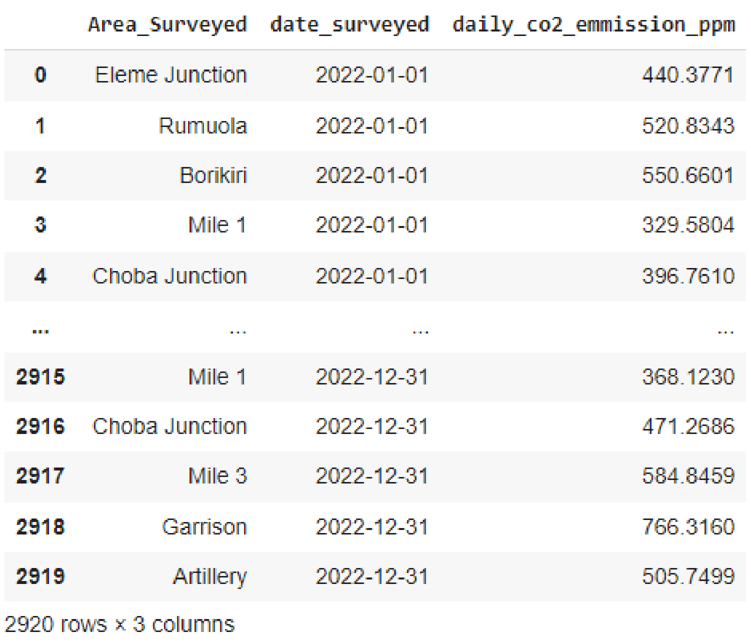
- Timestamps: Recorded as the survey date.
- CO_2 Concentration Levels: Measured in parts per million (ppm).
- Locations: Geographical areas in Port Harcourt where the data was obtained.

Critical Columns for Analysis

1. Location: Specific geographical areas in Port Harcourt, Rivers State, Nigeria, where measurements were taken, including:
   - Eleme Junction
   - Rumuola
   - Borikiri
   - Mile 1
   - Choba Junction
   - Mile 3
   - Garrison
   - Artillery
2. Date: The recorded dates during measurement, facilitating time-series analysis.
3. CO_2 ppm: The concentration of carbon dioxide in parts per million, which is the primary focus of the study.

Dataset Overview

- The dataset consists of 2,920 entries, enabling extensive statistical analysis and reliable forecasting.
- It captures typical variations in CO_2 levels influenced by specific emission sources.
- A snippet of the dataset is shown in Fig. 3 below.



	Area_Surveyed	date_surveyed	daily_co2_emmission_ppm
0	Eleme Junction	2022-01-01	440.3771
1	Rumuola	2022-01-01	520.8343
2	Borikiri	2022-01-01	550.6601
3	Mile 1	2022-01-01	329.5804
4	Choba Junction	2022-01-01	396.7610
...	...	...	...
2915	Mile 1	2022-12-31	368.1230
2916	Choba Junction	2022-12-31	471.2686
2917	Mile 3	2022-12-31	584.8459
2918	Garrison	2022-12-31	766.3160
2919	Artillery	2022-12-31	505.7499

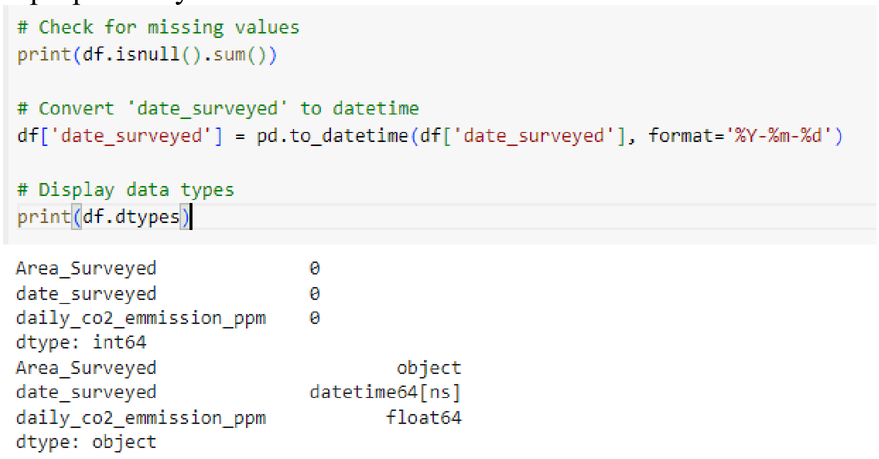
2920 rows × 3 columns

Figure 3: Dataset Overview

Data Cleaning

- Library Used: The Pandas library in Python was utilized for data cleaning and handling missing values.
- Data Cleaning Outcome:

- After applying data cleaning techniques, the dataset was confirmed to be clean with no missing values (as shown in Fig. 4).
- The data type of the date column was converted to a date-time format to ensure proper analysis.



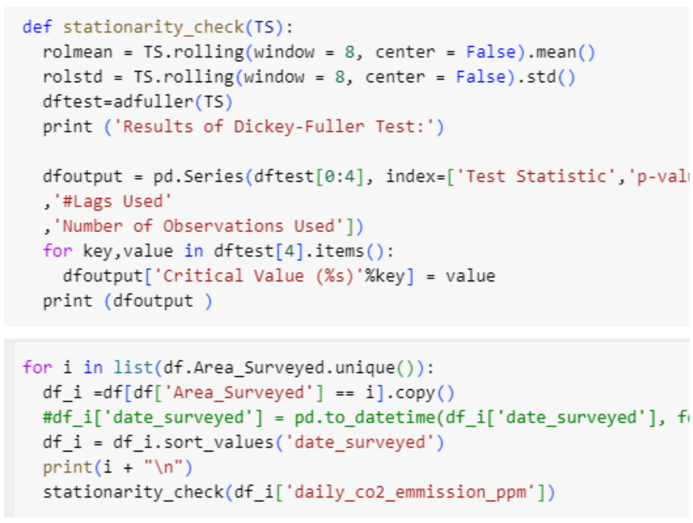
```
# Check for missing values
print(df.isnull().sum())

# Convert 'date_surveyed' to datetime
df['date_surveyed'] = pd.to_datetime(df['date_surveyed'], format='%Y-%m-%d')

# Display data types
print(df.dtypes)
```

```
Area_Surveyed              0
date_surveyed              0
daily_co2_emmission_ppm    0
dtype: int64
Area_Surveyed                      object
date_surveyed              datetime64[ns]
daily_co2_emmission_ppm           float64
dtype: object
```

Figure 4: Data Cleaning with Python

Stationarity Test

- Importance of Stationarity:
  - In time series analysis, stationarity is crucial as it ensures that the statistical properties of the dataset, such as mean and variance, remain consistent over time.
- Stationarity Validation:
  - To validate the stationarity of our dataset, we employed two widely recognized statistical tests:
  - Augmented Dickey-Fuller (ADF) Test: Checks for unit roots in the time series data, which would indicate non-stationarity.
  - Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test: Complements the ADF test by testing the null hypothesis that the time series is stationary around a deterministic trend.
- Test Results:
  - The dataset was found to be non-stationary and required transformation to achieve stationarity for model building.
  - The implementation of these tests in Python is illustrated in Fig. 5.

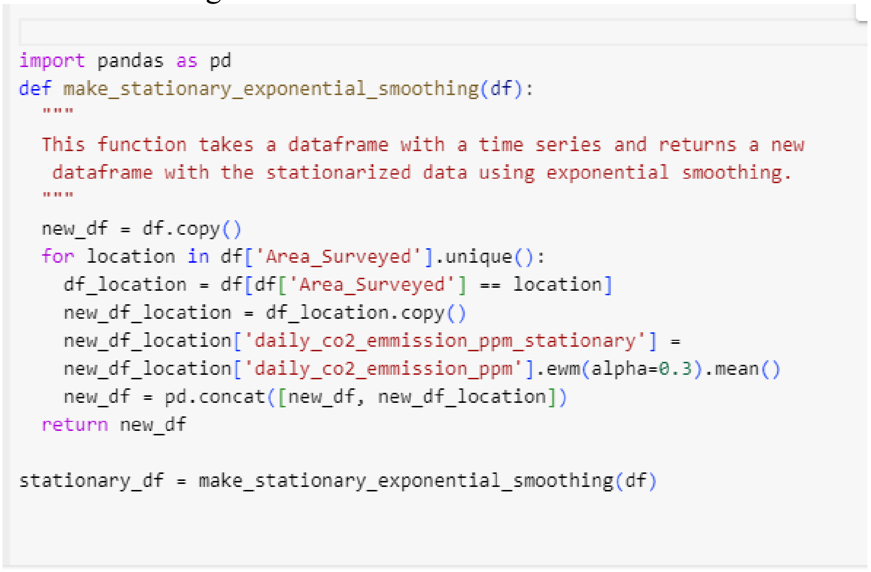


```
def stationarity_check(TS):
  rolmean = TS.rolling(window = 8, center = False).mean()
  rolstd = TS.rolling(window = 8, center = False).std()
  dftest=adfuller(TS)
  print ('Results of Dickey-Fuller Test:')

  dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-val
  ,'#Lags Used'
  ,'Number of Observations Used'])
  for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value
  print (dfoutput )
```

```
for i in list(df.Area_Surveyed.unique()):
  df_i =df[df['Area_Surveyed'] == i].copy()
  #df_i['date_surveyed'] = pd.to_datetime(df_i['date_surveyed'], f
  df_i = df_i.sort_values('date_surveyed')
  print(i + "\n")
  stationarity_check(df_i['daily_co2_emmission_ppm'])
```

Figure 5: Stationarity Test in Python

Converting Non-Stationary Data to Stationary
Goal: To ensure the time series data becomes stationary, making it suitable for time series analysis techniques.
Method:
- Exponential Smoothing: Applied to convert non-stationary data into stationary data.
  - Stationary data has a constant mean, variance, and autocorrelation structure over time.
  - The data from multiple locations was processed separately, ensuring the stationarity transformation was applied individually to each subset.
- Smoothing Technique:
  - Exponential smoothing assigns decreasing weights to past observations.
  - Alpha (Smoothing Factor): Controls the rate at which the weights decrease, with higher alpha values discounting older observations more rapidly.
  - This technique helps smooth out short-term fluctuations and highlights longer-term trends or cycles.
- Outcome: As shown in Fig. 6, the data was successfully converted to stationary using exponential smoothing.


```
import pandas as pd
def make_stationary_exponential_smoothing(df):
  """
  This function takes a dataframe with a time series and returns a new
   dataframe with the stationarized data using exponential smoothing.
  """
  new_df = df.copy()
  for location in df['Area_Surveyed'].unique():
    df_location = df[df['Area_Surveyed'] == location]
    new_df_location = df_location.copy()
    new_df_location['daily_co2_emmission_ppm_stationary'] =
    new_df_location['daily_co2_emmission_ppm'].ewm(alpha=0.3).mean()
    new_df = pd.concat([new_df, new_df_location])
  return new_df

stationary_df = make_stationary_exponential_smoothing(df)
```
Figure 6: Making Data Stationary with Python

Model Building
Dataset Split:
- The dataset was divided into 60:20:20 ratios for training, testing, and validation.

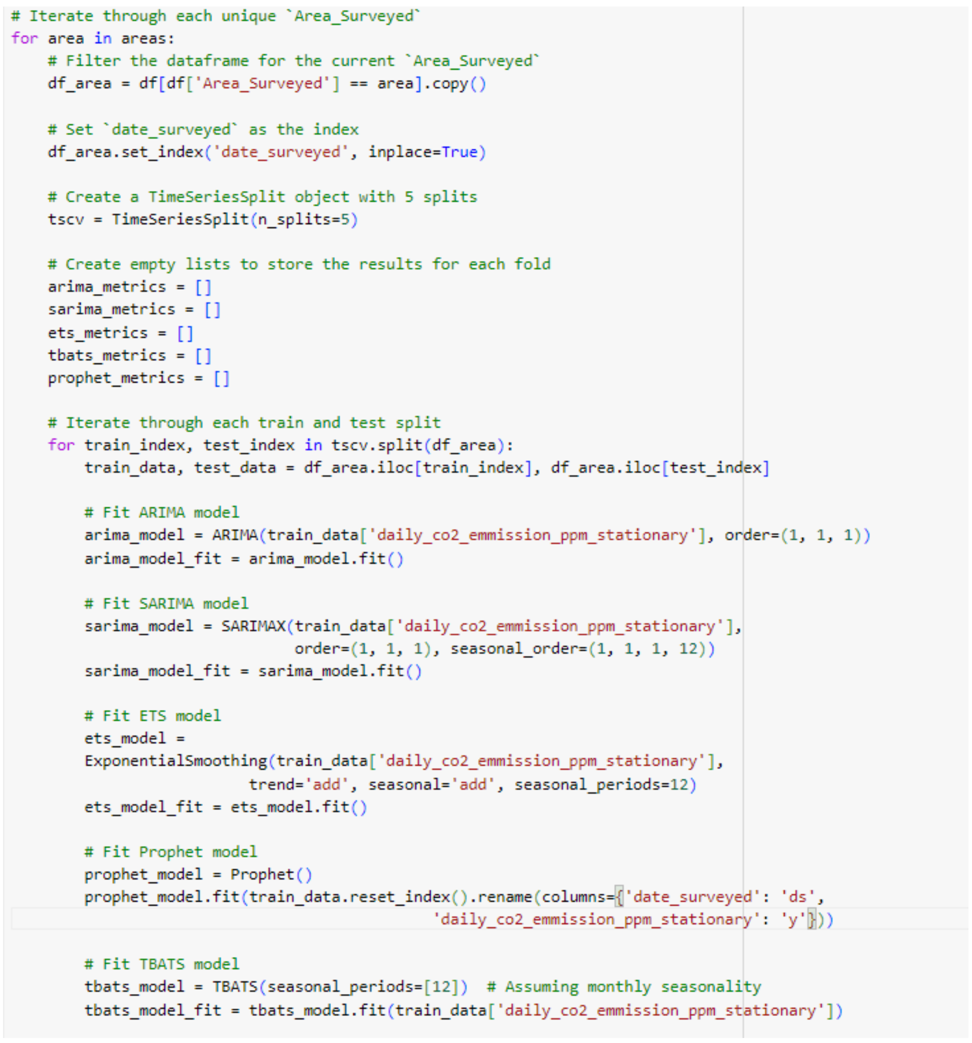
Parameter Specification Techniques:
1. Grid Search:
   - A systematic approach to exploring multiple combinations of parameter settings.
   - Cross-validation was used to identify the best-performing parameters.
2. Akaike Information Criterion (AIC):
   - A model selection criterion that assesses the goodness of fit while penalizing for the number of parameters, helping prevent over-fitting.

Models Evaluated:
1. ARIMA (Autoregressive Integrated Moving Average):
   - Captures autocorrelations but struggles with seasonality and non-linear patterns.
2. SARIMA (Seasonal ARIMA):
   - Extends ARIMA to handle seasonality by incorporating seasonal differencing.
3. ETS (Error, Trend, Seasonal):
   - Addresses error, trend, and seasonal components, providing flexibility in forecasting.

4. TBATS (Trigonometric Box-Cox ARMA Trend Seasonal):
   - Designed for complex seasonal patterns and non-linearity.
5. Prophet:
   - Developed by Facebook, this model handles seasonality and trends effectively and is robust to missing data.

Model Selection:
- ETS Model: Chosen for its ability to handle both trend and seasonal components and residual white noise in the data (Fig. 7).
- Parameter Optimization: Grid search and AIC were used to select the best parameters for the ETS model.

Model Comparison:
- The ETS model was compared with ARIMA, SARIMA, Prophet, and TBATS models using various metrics:
- Mean Absolute Error (MSE)
- Root Mean Squared Error (RMSE)
- Mean Absolute Percentage Error (MAPE)
- Accuracy



```
# Iterate through each unique `Area_Surveyed`
for area in areas:
    # Filter the dataframe for the current `Area_Surveyed`
    df_area = df[df['Area_Surveyed'] == area].copy()

    # Set `date_surveyed` as the index
    df_area.set_index('date_surveyed', inplace=True)

    # Create a TimeSeriesSplit object with 5 splits
    tscv = TimeSeriesSplit(n_splits=5)

    # Create empty lists to store the results for each fold
    arima_metrics = []
    sarima_metrics = []
    ets_metrics = []
    tbats_metrics = []
    prophet_metrics = []

    # Iterate through each train and test split
    for train_index, test_index in tscv.split(df_area):
        train_data, test_data = df_area.iloc[train_index], df_area.iloc[test_index]

        # Fit ARIMA model
        arima_model = ARIMA(train_data['daily_co2_emmission_ppm_stationary'], order=(1, 1, 1))
        arima_model_fit = arima_model.fit()

        # Fit SARIMA model
        sarima_model = SARIMAX(train_data['daily_co2_emmission_ppm_stationary'],
                               order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
        sarima_model_fit = sarima_model.fit()

        # Fit ETS model
        ets_model =
        ExponentialSmoothing(train_data['daily_co2_emmission_ppm_stationary'],
                             trend='add', seasonal='add', seasonal_periods=12)
        ets_model_fit = ets_model.fit()

        # Fit Prophet model
        prophet_model = Prophet()
        prophet_model.fit(train_data.reset_index().rename(columns={'date_surveyed': 'ds',
                                                   'daily_co2_emmission_ppm_stationary': 'y'}))

        # Fit TBATS model
        tbats_model = TBATS(seasonal_periods=[12])  # Assuming monthly seasonality
        tbats_model_fit = tbats_model.fit(train_data['daily_co2_emmission_ppm_stationary'])
```

Figure 7: Model Building with Python

Model Evaluation

Training and Validation:

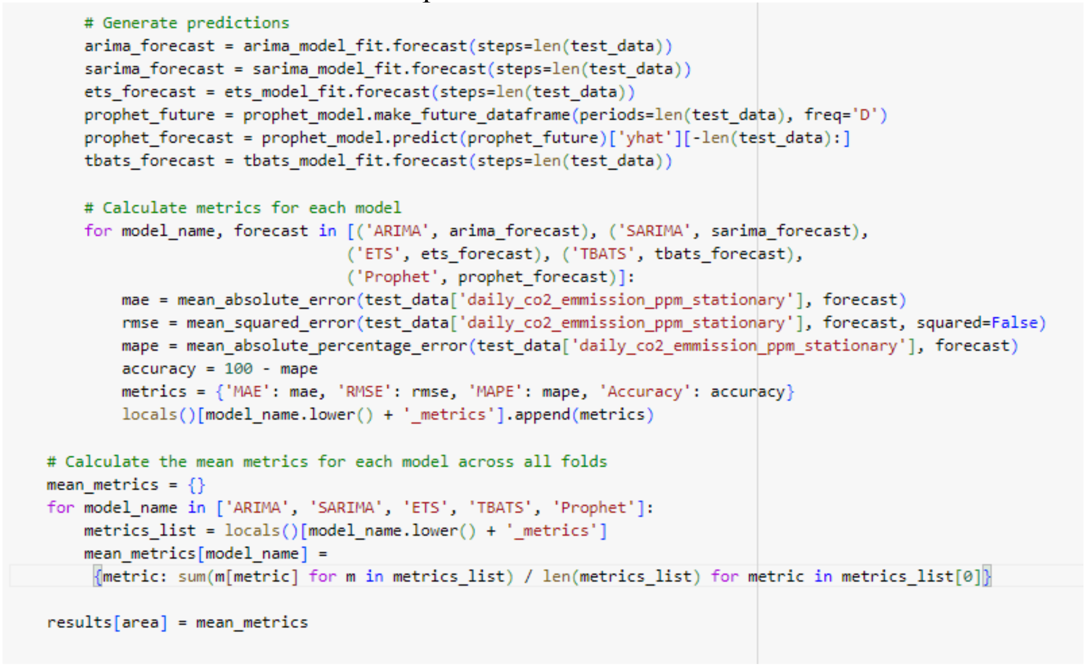
- Models were trained on historical data and validated on a separate set to assess their predictive capabilities.

Evaluation Metrics:

1. Mean Squared Error (MSE):
   - Measures the average squared difference between the predicted and actual values.
   - Formula: $\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|^2$
2. Root Mean Squared Error (RMSE):
   - The square root of MSE, measuring the average magnitude of the errors.
   - Formula: $\mathrm{RMSE} = \sqrt{\mathrm{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|^2}$
   - RMSE is sensitive to outliers, giving higher weight to significant errors.
3. Mean Absolute Percentage Error (MAPE):
   - Measures the average absolute percentage error between the actual and predicted values.
   - Formula: $\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n} | \frac{y_i - \hat{y}_i}{\hat{y}_i} | \times 100\%$
   - Expressed as a percentage, with lower values indicating better model accuracy.
4. Accuracy:
   - Measures the proportion of correct predictions made by the model.
   - Formula: $\mathrm{Accuracy} = \frac{\mathrm{TP+TN}}{\mathrm{total}}$
   - Higher accuracy indicates a greater proportion of correct predictions.

Implementation:

- Model evaluation was implemented using the Scikit-learn library, as shown in Fig. 8.
- The data was looped through various models and locations to gather metrics per model and location for effective comparison.



```python
        # Generate predictions
        arima_forecast = arima_model_fit.forecast(steps=len(test_data))
        sarima_forecast = sarima_model_fit.forecast(steps=len(test_data))
        ets_forecast = ets_model_fit.forecast(steps=len(test_data))
        prophet_future = prophet_model.make_future_dataframe(periods=len(test_data), freq='D')
        prophet_forecast = prophet_model.predict(prophet_future)['yhat'][-len(test_data):]
        tbats_forecast = tbats_model_fit.forecast(steps=len(test_data))

        # Calculate metrics for each model
        for model_name, forecast in [('ARIMA', arima_forecast), ('SARIMA', sarima_forecast),
                                     ('ETS', ets_forecast), ('TBATS', tbats_forecast),
                                     ('Prophet', prophet_forecast)]:
            mae = mean_absolute_error(test_data['daily_co2_emmission_ppm_stationary'], forecast)
            rmse = mean_squared_error(test_data['daily_co2_emmission_ppm_stationary'], forecast, squared=False)
            mape = mean_absolute_percentage_error(test_data['daily_co2_emmission_ppm_stationary'], forecast)
            accuracy = 100 - mape
            metrics = {'MAE': mae, 'RMSE': rmse, 'MAPE': mape, 'Accuracy': accuracy}
            locals()[model_name.lower() + '_metrics'].append(metrics)

    # Calculate the mean metrics for each model across all folds
    mean_metrics = {}
    for model_name in ['ARIMA', 'SARIMA', 'ETS', 'TBATS', 'Prophet']:
        metrics_list = locals()[model_name.lower() + '_metrics']
        mean_metrics[model_name] =
         {metric: sum(m[metric] for m in metrics_list) / len(metrics_list) for metric in metrics_list[0]}

    results[area] = mean_metrics
```

Figure 8: Model Evaluation with Python

RESULTS

Table 1: Table of Results of Model Comparison

Location	ARIMA MAE	ARIMA RMSE	SARIMA MAE	SARIMA RMSE	ETS MAE	ETS RMSE	TBATS MAE	TBATS RMSE	Prophet MAE	Prophet RMSE
Artillery	8.189929	9.414434	5.216727	6.434587	6.51589	7.745816	8.643169	9.87898	7.011866	8.023718
Borikiri	8.979762	10.19451	17.7717	20.46139	11.50804	13.19589	7.498978	8.707972	10.5312	12.19623
Choba Junction	8.602403	10.06618	3.967416	4.918523	4.865188	5.939162	7.37256	8.646515	5.651794	6.893803
Eleme Junction	26.00788	30.18519	17.49246	20.36282	17.81341	20.42345	16.91294	19.68541	15.80758	17.83459
Garrison	28.59987	34.67196	33.50029	40.94301	23.54225	28.73655	34.28987	41.60041	28.08073	33.88046
Mile 1	4.845608	5.689214	5.609525	6.64158	5.573555	6.474243	4.342932	5.189522	6.918413	8.029042
Mile 3	18.932	22.34101	29.793	35.9177	29.51151	35.51097	26.99706	32.71751	24.68793	30.23852
Rumuola	3.018585	3.744976	3.169828	3.913105	3.934054	4.959352	3.195726	3.90882	6.205812	7.437275
Average	13.397	15.78843	14.56512	17.44909	12.90799	15.37318	13.65665	16.29189	13.11192	15.5667

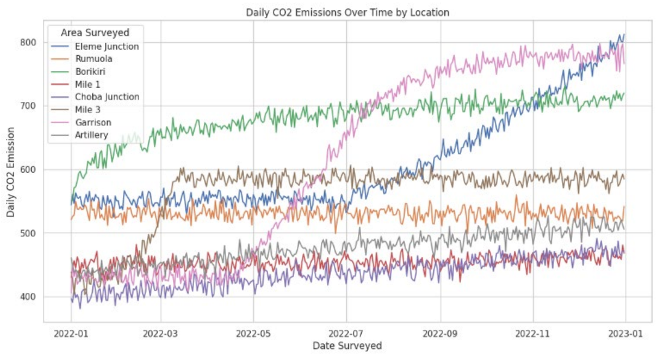


Figure 9: Plot of emissions over time

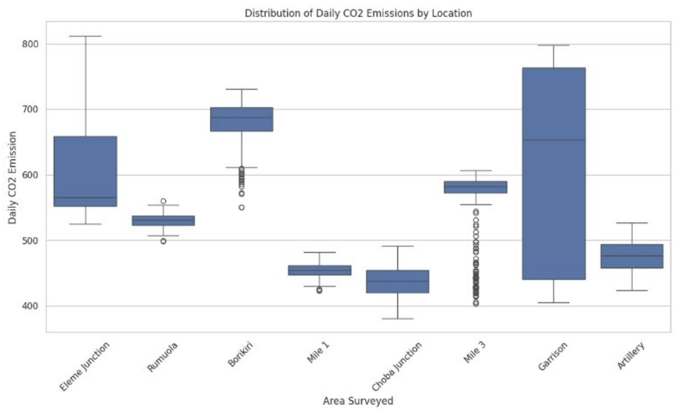
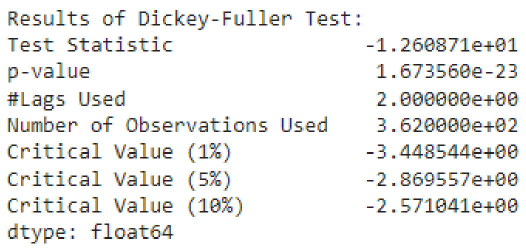


Figure 10: BoxPlot Of Emissions According to locations



```
Results of Dickey-Fuller Test:
Test Statistic                 -1.260871e+01
p-value                         1.673560e-23
#Lags Used                      2.000000e+00
Number of Observations Used     3.620000e+02
Critical Value (1%)            -3.448544e+00
Critical Value (5%)            -2.869557e+00
Critical Value (10%)           -2.571041e+00
dtype: float64
```

Figure 11: Stationarity test Results

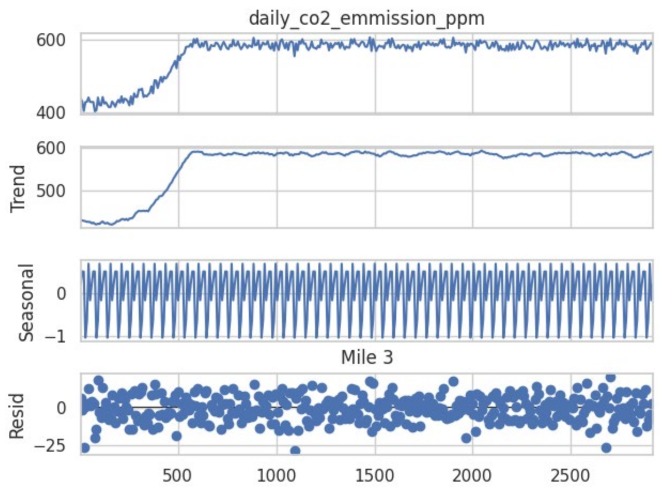


Figure 12: Seasonal Decomposition

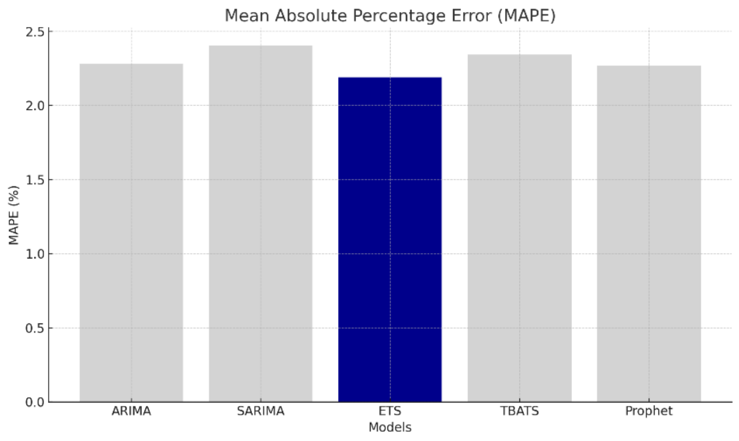


Figure 13: MAPE for Model Comparison

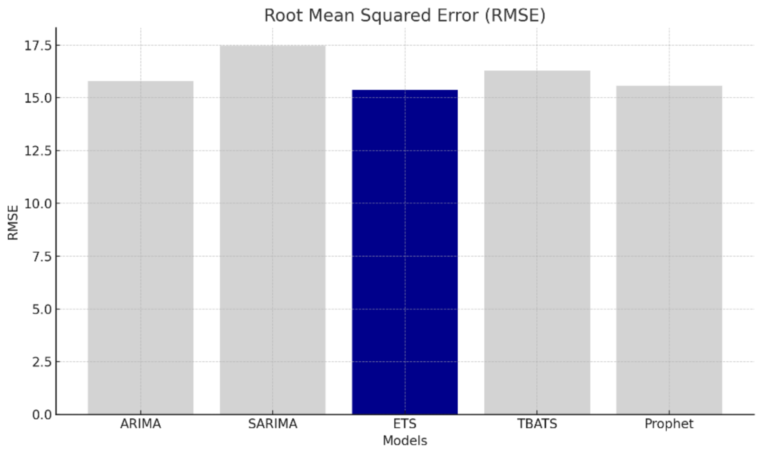


Figure 14: RMSE for Model Comparison

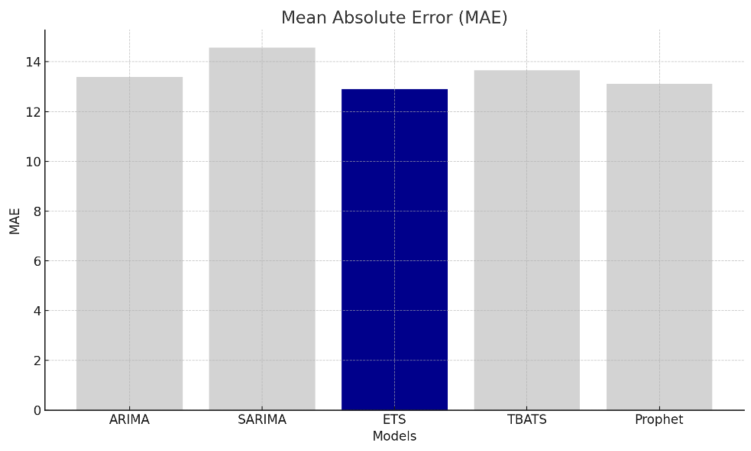


Figure 15: MAE for Model Comparisons

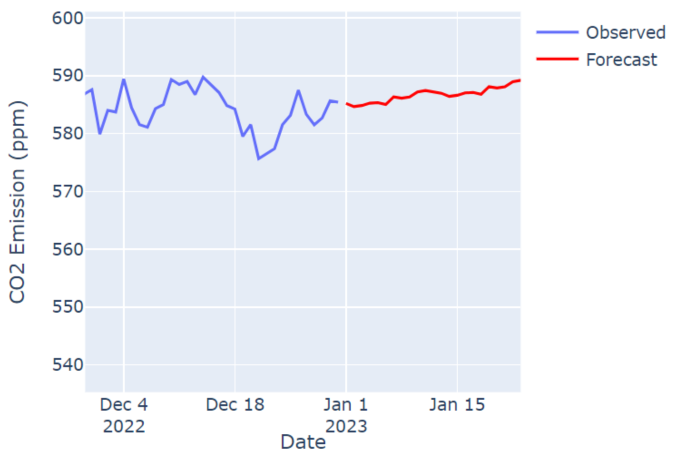


Figure 16: Forecast Example

DISCUSSION

The analysis of CO_2 emissions data revealed clear trends and patterns, with an upward trend observed across all the areas studied. This consistent increase in emissions over the years is likely due to a combination of factors, including ongoing industrial activities, population growth, and other human-driven activities that contribute to greenhouse gas emissions. The data underscores a crucial point: CO_2 emissions are continuing to rise.

When evaluating the seasonal decomposition plots alongside the CO_2 emissions data, it became evident that each study area exhibited similar seasonal patterns. These variations are likely influenced by factors such as weather changes, industrial production cycles, and other environmental conditions. Understanding these seasonal patterns is essential, as it allows us to identify when emissions typically increase during different times of the year.

In terms of model performance, a detailed analysis revealed that ARIMA and SARIMA models performed well in certain locations but struggled in others. Similar inconsistencies were observed with the TBATS and Prophet models. However, the ETS model consistently delivered strong performance across all locations, effectively capturing errors, seasonality, and trends better than the other models. While SARIMA was also effective in some cases, it performed poorly with data from specific locations.

The ETS model proved to be highly accurate in predicting future CO_2 emissions, with a mean absolute error (MAE) of 14.82 and a root mean square error (MSE) of 18.91, as detailed in Table 1. The model's performance was thoroughly evaluated using established metrics, as shown in Figures 12, 14, and 15. The model parameters were selected through grid search and validated using the Akaike Information Criterion (AIC) to ensure an optimal balance between model complexity and accuracy. When comparing ETS model predictions to actual CO_2 concentrations, the results were notably accurate, with predicted values falling within the 95% confidence interval of the differences between measured and predicted concentrations.

CONCLUSIONS

This work has significantly advanced environmental forecasting by rigorously evaluating and enhancing algorithms pertinent to predicting greenhouse gas emissions. We demonstrated that the limitations found in traditional approaches are often inherent and that advanced forecasting methods, such as ETS and TBATS, offer substantial improvements in modeling greenhouse gas time series. These methods are particularly effective in capturing complex seasonal patterns and non-linear dynamics while maintaining a comparable level of complexity. Conducting a comparative analysis of different models before selecting the final one for deployment is crucial to ensuring accurate, data-driven predictions.

Beyond the immediate goal of modeling complex climate phenomena, there is a compelling need for robust and effective environmental forecasting models like the ones discussed here. In an era where climate science increasingly influences political discourse—demanding scientifically informed debates among decision-makers about the best strategies for addressing climate challenges—predictive models will be instrumental in shaping policy. Moving forward, the priority should be to further refine these models, optimizing the balance between computational efficiency and predictive accuracy. This optimization will enable real-time deployment, making these models a valuable tool for policymakers and environmental scientists aiming to mitigate the effects of climate change through informed, timely interventions.

REFERENCES

[1] Adhikari, R., & Agrawal, R. (2013). An introductory study on time series modelling and forecasting. https://doi.org/10.13140/2.1.2771.8084

[2] Al-Turaiki, I., Almutlaq, F., Alrasheed, H., & Alballa, N. (2021). Empirical evaluation of alternative time-series models for COVID-19 forecasting in Saudi Arabia. *International Journal of Environmental Research and Public Health, 18*(16), 8660.

[3] Andreozzi, L., Blaconá, M. T., & Magnano, L. (2014). Time series models for different seasonal patterns. In *The 34th International Symposium on Forecasting (ISF 2014)*. International Institute of Forecasters, Rotterdam.

[4] Ayodele, T. (2010). Machine learning overview. https://doi.org/10.5772/9374

[5] Bodily, S., & Weatherford, L. (2006). Seasonality in time series forecasting. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.911866

[6] British Geological Survey. (2023, April 5). The greenhouse effect. British Geological Survey. https://www.bgs.ac.uk/discovering-geology/climate-change/how-does-the-greenhouse-effect-work/

[7] Darkwah, W. K., Odum, B., Addae, M., Koomson, D., Kwakye Danso, B., Oti-Mensah, E., Asenso, T., & Buanya, B. (2018). Greenhouse effect: Greenhouse gases

and their impact on global warming. *Journal of Scientific Research and Reports, 17*, 1-9. https://doi.org/10.9734/JSRR/2017/39630

[8] De Myttenaere, A., Golden, B., Le Grand, B., & Rossi, F. (2016). Mean absolute percentage error for regression models. https://doi.org/10.1016/j.neucom.2015.12.114

[9] De Livera, A. M., Hyndman, R. J., & Snyder, R. D. (2011). Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association, 106*(496), 1513–1527. https://doi.org/10.1198/JASA.2011.TM09771

[10] Fildes, R. (1986). Forecasting trends in time series. *International Journal of Forecasting, 2*(3), 383–384. https://doi.org/10.1016/0169-2070(86)90056-7

[11] Gardner, E. S., & McKenzie, E. (1989). Note—Seasonal exponential smoothing with damped trends. *Management Science, 35*(3), 372–376. https://doi.org/10.1287/MNSC.35.3.372

[12] Holt, C. C. (2004). Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting, 20*(1), 5–10. https://doi.org/10.1016/J.IJFORECAST.2003.09.015

[13] International Energy Agency, I. (2023a). CO 2 Emissions in 2023. www.iea.org

[14] Jain, G., & Mallick, B. (2017). A study of time series models ARIMA and ETS. *SSRN Electronic Journal*. https://doi.org/10.2139/SSRN.2898968

[15] Kumar Dubey, A., Kumar, A., García-Díaz, V., Kumar Sharma, A., & Kanhaiya, K. (2021). Study and analysis of SARIMA and LSTM in forecasting time series data. *Sustainable Energy Technologies and Assessments, 47*, 101474. https://doi.org/10.1016/J.SETA.2021.101474

[16] Larsson, S. (2002). Bayesian methods and evidence. In *Perspectives in Neural Computing* (pp. 281-300). Springer. https://doi.org/10.1007/978-1-4471-0151-2_15

[17] Marriott, J., & Newbold, P. (1998). Bayesian comparison of ARIMA and stationary ARMA models. *International Statistical Review, 66*(3), 323–336. https://doi.org/10.1111/J.1751-5823.1998.TB00376.X

[18] Momin, B., & Chavan, G. (2018). Univariate time series models for forecasting stationary and non-stationary data: A brief review. In *Smart Innovation, Systems and Technologies* (Vol. 84, pp. 219–226). https://doi.org/10.1007/978-3-319-63645-0_24

[19] Nutongkaew, P., Waewsak, J., Chaichana, T., & Gagnon, Y. (2014). Greenhouse gases emission of refuse derived fuel-5 production from municipal waste and palm kernel. *Energy Procedia, 56*, 219-226. https://doi.org/10.1016/j.egypro.2014.07.087

[20] Staff, C. (2024, March 27). What is machine learning? Definition, types, and examples. Coursera. https://www.coursera.org/articles/what-is-machine-learning

[21] Surakhi, O., Zaidan, M. A., Fung, P. L., Hossein Motlagh, N., Serhan, S., AlKhanafseh, M., Ghoniem, R. M., & Hussein, T. (2020). Time-Lag Selection for Time-Series Forecasting Using Neural Network and Heuristic Algorithm. *Electronics, 10*(20), 2518. https://doi.org/10.3390/electronics10202518

[22] Smith, P., Reay, D., & Smith, J. (2021). Agricultural methane emissions and the potential formitigation. *Philosophical Transactions of the Royal Society A*, *379*(2210), 20200451.

[23] Taylor, S., & Letham, B. (2017). Forecasting at scale. https://doi.org/10.7287/peerj.preprints.3190v2

[24] United Nations Environment Programme. (n.d.). Methane emissions are driving climate change. Here's how to reduce them. UNEP. https://www.unep.org/news-and-stories/story/methane-emissions-are-driving-climate-change-heres-how-reduce-them

[25] Wang, S., Li, C., & Lim, A. (2019). Why are the ARIMA and SARIMA not sufficient? Retrieved from http://arxiv.org/abs/1904.07632

[26] Wang, W., & Lu, Y. (2018). Analysis of the mean absolute error (MAE) and the root mean square error (RMSE) in assessing rounding model. In *IOP Conference Series: Materials Science and Engineering* (Vol. 324, p. 012049). https://doi.org/10.1088/1757-899X/324/1/012049

[27] Xie, T., & Ding, J. (2020). Forecasting with multiple seasonality. http://arxiv.org/abs/2008.12340