International Centre for Education and Research (ICER)
VIT-Bangalore

Predicting Climate Change Impact on the World.

CS6510 – PROJECT 1

REPORT

Submitted by

SHIVAM-24MSP3080

In partial fulfilment for the award of the degree of

POST GRADUATE PROGRAMME

INTERNATIONAL CENTRE FOR HIGHER EDUCATION AND RESEARCH

VIT BANGALORE

DECEMBER, 2024

International Centre for Education and Research (ICER)
VIT-Bangalore

BONAFIDE CERTIFICATE

Certified that this project report **"Predicting Climate Change Impact on the World."** is the Bonafide record of work done by **"SHIVAM-24MSP3080"** who carried out the project work under my supervision.

Signature of the Supervisor	**Signature of Director**
Dr. Pitchumani Angayarkanni S	**Prof. Prema M**
Professor,	**Director,**
ICER	ICER
VIT Bangalore	VIT Bangalore.

Evaluation Date: 13-12-2024

International Centre for Education and Research (ICER)
VIT-Bangalore

ACKNOWLEDGEMENT

I express my sincere gratitude to our director of ICER ***Prof. Prema M.*** for their support and for providing the required facilities for carrying out this study.

I wish to thank my faculty supervisor ***Dr. Pitchumani Angayarkanni S***, Professor, ICER for extending help and encouragement throughout the project. Without her continuous guidance and persistent help, this project would not have been a success for me.

I am grateful to all the members of ICER, my beloved parents, and friends/Batchmates for extending the support, who helped me to overcome obstacles in the study.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	
	LIST OF FIGURES	
	LIST OF TABLES	
1	**INTRODUCTION**	**1-2**
	1.1. Research Topic	1
	1.2 Background and Rationale	1-2
	1.3 Scope and Limitations	3
2	**LITERATURE REVIEW**	**3-10**
3	**OBJECTIVE**	**11**
4	**PROPOSED METHODOLOGY**	**12-14**
5	**TOOLS AND TECHNIQUES**	**15-17**
6	**IMPLEMENTATION (DATABASE WISE)**	**32-52**
7	**RESULTS AND DISCUSSIONS**	**53-57**
8	**CONCLUSION**	**58**
9	**FUTURE ENHANCEMENT**	**59**
10	**APPENDICES INDEX**	**60**
	Appendix-1: Code – Full Code (DATABASE WISE)	**61-124**
	Appendix-2: Plagiarism Report	**125**
11	**REFERENCES**	**126**
12	**WORKLOG**	**127-130**

ABSTRACT

Research Question: This study focuses on developing predictive models for key climate change parameters in India, including Atmospheric CO2 Concentrations, Change in Mean Sea Levels, Forest Land Cover, and Annual Surface Temperature Change. The goal is to understand the ongoing effects of global warming and support the development of climate policies and strategies to mitigate these impacts especially on INDIA.

Methodology:

1. Data Collection: Historical climate data was sourced from the International Monetary Fund (IMF) Climate Data Platform.
2. Data Preprocessing: The data was cleaned, normalized, and transformed to make it suitable for analysis.
3. Exploratory Data Analysis (EDA): Trends, patterns, and correlations in the data were identified through visualizations and statistical analysis.
4. Model Development: Various models, including Linear Regression, Multiple Linear Regression ARIMA, SARIMA and LSTM were developed and compared.
5. Training and Validation: Metrics like MSE, MAE, RMSE, and R-squared were used for evaluation.
6. Scenario Analysis: Different climate scenarios were developed to understand potential future states and their impacts on ecosystems, human health, and the economy in India.

Results:

- Model Performance: The LSTM model showed superior performance in capturing long-term dependencies and complex patterns in the data, making it well-suited for long-term predictions while ARIMA is well-suited for short term predictions.
- Scenario Insights: The scenario analysis provided valuable insights into the potential impacts of climate change on India's ecosystems, public health, and economy under different future states.

Conclusion:

- The study successfully developed predictive models for key climate change parameters in India.
- The findings highlight the importance of using advanced machine learning techniques to improve prediction accuracy.

- The insights gained from the scenario analysis can inform climate policies and strategies, helping India mitigate and adapt to the adverse effects of climate change.
- This research fills a critical gap in region-specific climate studies and supports global efforts in combating climate change.

LIST OF FIGURES

Figure No.	Figure Name	Pg. No.
Fig. 4.1	Methodology	12
Fig. 6.1	EDA Annual Mean Surface Temp	19
Fig. 6.2	Linear Regression Model	19
Fig. 6.3	Prediction using Linear Regression	20
Fig. 6.4	ACF and PACF Plots	21
Fig. 6.5	ARIMA model building	22
Fig. 6.6	Predicting using ARIMA model	23
Fig. 6.7	LSTM model	23
Fig. 6.8	Predicting using LSTM model	24
Fig. 6.9	EDA Atmospheric co2 concentration	26
Fig. 6.10	Linear Regression Model	26

Fig. 6.11	Prediction using Linear Regression	27
Fig. 6.12	ACF and PACF Plots	28
Fig. 6.13	ARIMA model building	29
Fig. 6.14	LSTM model	30
Fig. 6.15	Predicting using LSTM model	30
Fig. 6.16	EDA Change in mean sea level	33
Fig. 6.17	Water Bodies surrounding INDIA	33
Fig. 6.18	Change in mean sea level for relevant water bodies	34
Fig. 6.19	Linear Regression Model	34
Fig. 6.20	Prediction using Linear Regression	35
Fig. 6.21	ACF and PACF Plots Indian ocean	38
Fig. 6.22	ACF and PACF Plots Bay of Bengal	39
Fig. 6.23	ACF and PACF Plots Arabian Sea	40
Fig. 6.24	ARIMA model building Indian ocean	41
Fig. 6.25	Predicting using ARIMA model	42
Fig. 6.26	ARIMA model building Bay of Bengal	43
Fig. 6.27	Predicting using ARIMA model	43

Fig. 6.28	ARIMA model building Arabian Sea	44
Fig. 6.29	Predicting using ARIMA model	45
Fig. 6.30	Predicting using LSTM model	45
Fig. 6.31	EDA Forest Area Cover	48
Fig. 6.32	Linear Regression Model	49
Fig. 6.33	Prediction using Linear Regression	50
Fig. 6.34	ACF and PACF Plots	50
Fig. 6.35	Predicting using ARIMA model	52
Fig. 6.36	LSTM model	52

LIST OF TABLES

Table No.	Table Name	Pg. No.
Table. 6.1	AUTO ARIMA Annual Mean Surface Temp	22
Table 6.2	LSTM Summary Annual Mean Surface Temp	24
Table. 6.3	AUTO ARIMA CO2 concentration	29
Table 6.4	LSTM Summary CO2 concentration	31
Table. 6.5	OLS Summary Indian Ocean	36
Table 6.6	OLS Summary Arabian Sea	36
Table. 6.7	OLS Summary Bay of Bengal	37
Table 6.8	AUTO ARIMA Indian Ocean	41
Table. 6.9	AUTO ARIMA Bay of Bengal	42
Table 6.10	AUTO ARIMA Arabian Sea	44
Table. 6.11	LSTM Summary Indian Ocean	46
Table 6.12	LSTM Summary Bay of Bengal	46
Table. 6.13	LSTM Summary Arabian Sea	46
Table 6.14	AUTO ARIMA Forest Area	51

Table. 6.15 LSTM summary Forest Area 52

Table 6.16 ARIMA vs LSTM 53

CHAPTER 1

INTRODUCTION

1.1 Research Topic:

This study focuses on developing predictive models for key climate change parameters for ***India***, including Atmospheric CO2 Concentrations, Change in Mean Sea Levels, Forest Land Cover, and Annual Surface Temperature Change. The goal is to understand the ongoing effects of global warming and support the development of climate policies and strategies to mitigate these impacts.

1.2 Background and Rationale:

Research Questions and Hypotheses: Climate change is profoundly affecting India's diverse ecosystems, human health, and economic stability. Despite the global significance of this issue, there is a noticeable gap in region-specific studies that address India's unique climate dynamics. Given India's vast and varied geography, from the Himalayas to coastal areas and arid regions to dense forests, it is crucial to understand how climate change impacts these diverse environments. Accurate predictions of climate parameters can help formulate effective policies and adaptive strategies, ensuring sustainable development and resilience against climate-related challenges.

- **Research Questions**:
    1. How will atmospheric CO2 concentrations in India change over the next few decades?
    2. What are the projected changes in mean sea levels along India's coastlines?
    3. How will forest land cover in India be affected by climate change?
    4. What are the expected trends in annual surface temperature change in India?
- **Hypotheses**:
    1. Atmospheric CO2 concentrations in India will continue to rise due to ongoing industrialization and urbanization.
    2. Mean sea levels along India's coastlines will increase, leading to higher risks of coastal flooding and erosion.
    3. Forest land cover in India will decrease due to deforestation and climate-induced changes in vegetation patterns.

4. Annual surface temperatures in India will show an upward trend, contributing to more frequent and severe heatwaves.

1.3 Scope and Limitations:

- **Scope**:
  - This research focuses on historical climate data specific to India sourced from the International Monetary Fund (IMF) Climate Data Platform.
  - The study aims to develop and validate predictive models using advanced machine learning and time-series forecasting techniques.
  - The analysis will include scenario planning to understand potential future states and their impacts on India's ecosystems, public health, and economy.

- **Limitations**:
  - The accuracy of predictions depends on the quality and completeness of the available data.
  - The models may not fully capture all regional climate variations and localized impacts.
  - The study's findings are limited to the specific parameters and timeframes analysed, and may not account for unforeseen climate events or interventions.

2. LITERATURE REVIEW

Climate Change Indicators used:

1. **Atmospheric CO2 Concentrations:**
    - Current Knowledge: Atmospheric CO2 levels are rising due to human activities such as burning fossil fuels, deforestation, and industrial processes.
    - Impact: CO2 is a significant greenhouse gas contributing to global warming and climate change.
2. **Change in Mean Sea Levels:**
    - Current Knowledge: Sea levels are increasing globally due to the melting of polar ice caps and glaciers and the thermal expansion of seawater as it warms.
    - Impact: Rising sea levels threaten coastal communities, ecosystems, and infrastructure through increased flooding and erosion.
3. **Forest Land Cover:**
    - Current Knowledge: Forest land cover is declining due to deforestation, land-use changes, and climate-related stress.
    - Impact: Forests are crucial for carbon sequestration, biodiversity, and climate regulation.
4. **Annual Surface Temperature Change:**
    - Current Knowledge: Global surface temperatures are rising, resulting in more frequent and intense heatwaves.
    - Impact: Increasing temperatures affect human health, agriculture, and natural ecosystems.

Models:

- Time-Series Forecasting Models: Linear Regression, ARIMA, LSTM used to analyse and predict trends in climate-related time series data.
- Comparative analysis between ARIMA and LSTM which performs better in short-term and long-term predictions.

Previous Studies:

- Numerous global studies have analysed climate data to identify trends in CO2 concentrations, sea level rise, deforestation, and temperature changes.
- Some regional studies have focused on specific areas but often lack the comprehensive approach required for understanding India's diverse climate dynamics and researches are quite old.

Gaps in Existing Literature:

1. **Regional Focus:**
    - Gap: Limited research specifically addressing climate change impacts in India, given its unique environmental and socio-economic context.
    - Contribution: This study aims to provide a comprehensive analysis focused on India, highlighting localized impacts and trends.
2. **Integrated Approach:**
    - Gap: Many studies analyse climate indicators in isolation rather than an integrated manner.
    - Contribution: This project will integrate multiple indicators (CO2 concentrations, sea levels, forest cover, temperature) to offer a holistic understanding of climate change impacts in India.
3. **Advanced Predictive Techniques:**
    - Gap: There is a lack of studies employing advanced machine learning and deep learning techniques for climate predictions in India.

- Contribution: This research will utilize advanced machine learning models like LSTM and ARIMA to improve prediction accuracy and provide more detailed insights.

4. **Scenario Planning:**
    - Gap: Insufficient use of scenario analysis to explore different future climate states and their potential impacts.
    - Contribution: The project will develop various climate scenarios to understand and plan for potential future impacts on India's ecosystems, health, and economy.

[1] The paper "***Tackling Climate Change with Machine Learning***," published on November 5, 2019, is a collaborative effort involving researchers from multiple prestigious institutions, including MIT, Cornell, Stanford, Harvard, DeepMind, Google AI, Microsoft Research, University of Pennsylvania, and Carnegie Mellon University. The paper is available as a preprint on arXiv with the number 1906.05433.

The research encompasses a comprehensive analysis of machine learning applications across various domains of climate change. It evaluates multiple algorithm types, including supervised learning, unsupervised learning, reinforcement learning, and deep learning approaches. Key performance metrics such as prediction accuracy, computational efficiency, data requirements, scalability, and real-world applicability are assessed.

One of the significant research limitations identified is the challenge of data quality and availability, along with computational resource constraints and model generalizability issues. The complexity of climate systems further complicates these challenges. Interdisciplinary collaboration challenges are also highlighted, including communication barriers between domains and the need to integrate

expertise from computer science, climate science, environmental policy, engineering, and economics.

The paper discusses scalability and implementation issues, emphasizing the need to translate research into practical solutions. Infrastructure adaptation requirements, the cost of large-scale implementation, and the integration of technical and policy measures are crucial considerations. Developing flexible machine learning models that can adapt to diverse environmental contexts, create transferable methodologies, and address regional variability is a key focus.

High-impact problem exploration areas include renewable energy optimization, carbon emissions reduction, climate modeling, ecosystem monitoring, urban planning, and agricultural innovation. The paper promotes collaborative research frameworks, encourages cross-domain knowledge exchange, and develops integrated research methodologies.

Recommendations for the future include continued investment in interdisciplinary research, developing more robust and adaptable machine learning models, enhancing data collection and sharing mechanisms, and creating policy frameworks that support technological innovation. Potential future approaches involve advanced AI-driven climate prediction models, more sophisticated emissions tracking systems, enhanced renewable energy grid management, and improved climate adaptation strategies.

The research emphasizes machine learning's transformative potential in addressing climate change while acknowledging the complex and multifaceted nature of environmental challenges.

[2] The paper titled "***A Novel Hybrid Machine Learning Model for Prediction of CO2 Using Socio-Economic and Energy Attributes for Climate Change Monitoring and Mitigation Policies,***" authored by Sachin Kumar, was published in Science Direct's Ecological Informatics, Volume 77, under Article Number 102253 in November 2023.

This research employs several machine learning techniques including Linear Regression (LR), Support Vector Regression (SVR), Random Forest Regressor (RFR), hybrid models, and deep learning models like Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and TabNet. The performance of these models was evaluated using various metrics such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and Symmetric Mean Absolute Percentage Error (SMAPE), with respective values of 0.0307, 0.0346, 5.1447, and 5.2267.

The study also identifies several data limitations, including the complexity of variables, issues with generalizability, the need for real-time data incorporation, exploration of additional variables, application to different regions, and the development of more advanced models. These challenges underline the complexities of accurately predicting CO2 emissions and the necessity for continuous improvement in model development.

The key observations highlight the research's focus on creating a hybrid machine learning approach to predict CO2 emissions using socio-economic and energy-related attributes. This model aims to assist in climate change monitoring and policy development. The relatively low error rates indicate promising predictive capabilities, although the authors note several limitations that future research could address to enhance the model's effectiveness.

The study underscores the potential of hybrid machine learning models in climate change monitoring and the critical need for interdisciplinary efforts to refine these models and improve their application in diverse environmental contexts.

[3] The paper titled "***ForestNet: Classifying Drivers of Deforestation in Indonesia using Deep Learning on Satellite Imagery***" was authored by Irvin et al. (2020) and updated by Ramachandran et al. (2024), with contributions from Stanford University, Descartes Labs, and RTI International. Initially published on November 11, 2020, the latest update was made on May 1, 2024.

This study focuses on classifying the drivers of deforestation in Indonesia using deep learning models applied to satellite imagery analysis. The research is particularly noted for its development of deep learning models and achieving superior classification accuracy.

Despite its strengths, the research identifies several limitations, including dataset constraints, model complexity, potential overfitting, and the lack of real-time analysis capabilities. Addressing these limitations is critical for improving the accuracy and applicability of the models.

Future research recommendations emphasize the need to enhance model accuracy, expand the dataset, and develop real-time monitoring capabilities. These steps are essential for creating more effective tools to combat deforestation.

In essence, the study offers an innovative approach to understanding deforestation patterns through advanced machine learning techniques. By concentrating on Indonesia, it aims to provide valuable insights into the drivers of forest loss, which can inform environmental conservation and policy-making efforts.

[4] The paper titled "***Sea Level Prediction Using Machine Learning***" by Rifat Tur, Erkin Tas, Ali Torabi Haghighi, and Ali Danandeh Mehr was published by MDPI on December 13, 2021. It was part of the special issue on the application of data pre- and post-processing methods for modeling hydro-climatological processes.

This study employs a machine learning approach with Multiple Linear Regression (MLR) as the primary technique. The performance of the model was evaluated using Root Mean Squared Error (RMSE) and Nash-Sutcliffe Efficiency (NSE) metrics. The results showed an RMSE value of 0 and an NSE of less than 0, indicating significant challenges in the model's predictive accuracy. These metrics suggest that the model's predictions may not effectively capture the variability in sea level measurements, potentially due to issues like oversimplification of the model, data preprocessing problems, or the unsuitability of the chosen machine learning technique for this specific prediction task.

The study identifies several key challenges and limitations, including the lack of real-time analysis, the need for improved model accuracy, and limited real-time monitoring capabilities. These challenges highlight the complexity of accurately predicting sea levels and the necessity for more sophisticated approaches.

The research underscores the importance of machine learning techniques for predicting sea levels, which is crucial for understanding the impacts of climate change, coastal management, and environmental planning. However, the performance metrics indicate that the current model may not be fully effective. Therefore, future research should explore more advanced machine learning models, improve data preprocessing techniques, develop real-time monitoring capabilities, and enhance model accuracy through more sophisticated approaches.

This study represents an important step in utilizing machine learning for environmental monitoring and highlights the need for ongoing research and development to address the identified challenges and limitations.

[5] The paper titled "***Tackling Extreme Urban Heat: A Machine Learning Approach to Assess the Impacts of Climate Change and the Efficacy of Climate Adaptation Strategies in Urban Microclimates***" was authored by Grant Buster, Jordan Cox, Brandon N. Benton, and Ryan N. King. These researchers are affiliated with the Strategic Energy Analysis Centre (SEAC), the National Renewable Energy Laboratory (NREL), and the Computational Science Centre (CSC). The publication date is November 8, 2024.

The study employs a multistep modeling process using a machine learning approach to understand and predict urban heat patterns, which is critical for addressing climate change impacts in urban environments. The performance of the models was evaluated using Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) metrics. Key research challenges identified include dataset limitations and the need for model accuracy enhancement.

This research is particularly timely, addressing the growing challenge of extreme urban heat – a critical issue in climate change adaptation. Urban areas are significantly more vulnerable to heat stress due to factors such as the urban heat island effect, concentrated built environments, limited green spaces, and high levels of human activity and infrastructure density.

The study's limitations suggest the need for more comprehensive data collection, improved data preprocessing techniques, expansion of the existing dataset, and continuous model refinement. These steps are essential to enhance the model's accuracy and applicability in predicting urban heat patterns.

The potential implications of this research are vast, providing valuable insights for urban planning, climate adaptation strategies, mitigation of heat-related health risks, infrastructure development, and energy management in urban environments. By understanding and predicting urban heat patterns more accurately, this research can inform effective climate adaptation and mitigation strategies, contributing to more resilient urban environments.

3. OBJECTIVE

The primary objective of this project is to develop accurate predictive models for key climate change parameters in India, specifically Atmospheric CO2 Concentrations, Change in Mean Sea Levels, Forest Land Cover, and Annual Surface Temperature Change. This involves:

1. **Understanding Trends**: Analyzing historical climate data to identify trends and patterns specific to India.
2. **Predictive Modeling**: Employing advanced machine learning techniques, including Linear Regression, Multiple Linear Regression, ARIMA, and LSTM models, to forecast future values of these climate parameters.
3. **Impact Assessment**: Evaluating the potential impacts of predicted changes on India's ecosystems, human health, and economic stability.
4. **Scenario Analysis**: Developing various climate scenarios to understand potential future states and their implications.
5. **Policy Support**: Providing valuable insights that can inform and shape effective climate policies and adaptive strategies tailored to India's unique context.
6. **Awareness and Preparedness**: Enhancing understanding and awareness of climate change impacts among stakeholders, thereby promoting informed decision-making and preparedness.

Through this project, we aim to contribute to the global effort to mitigate and adapt to climate change, ensuring a sustainable and resilient future for India.

4. PROPOSED METHODOLOGY

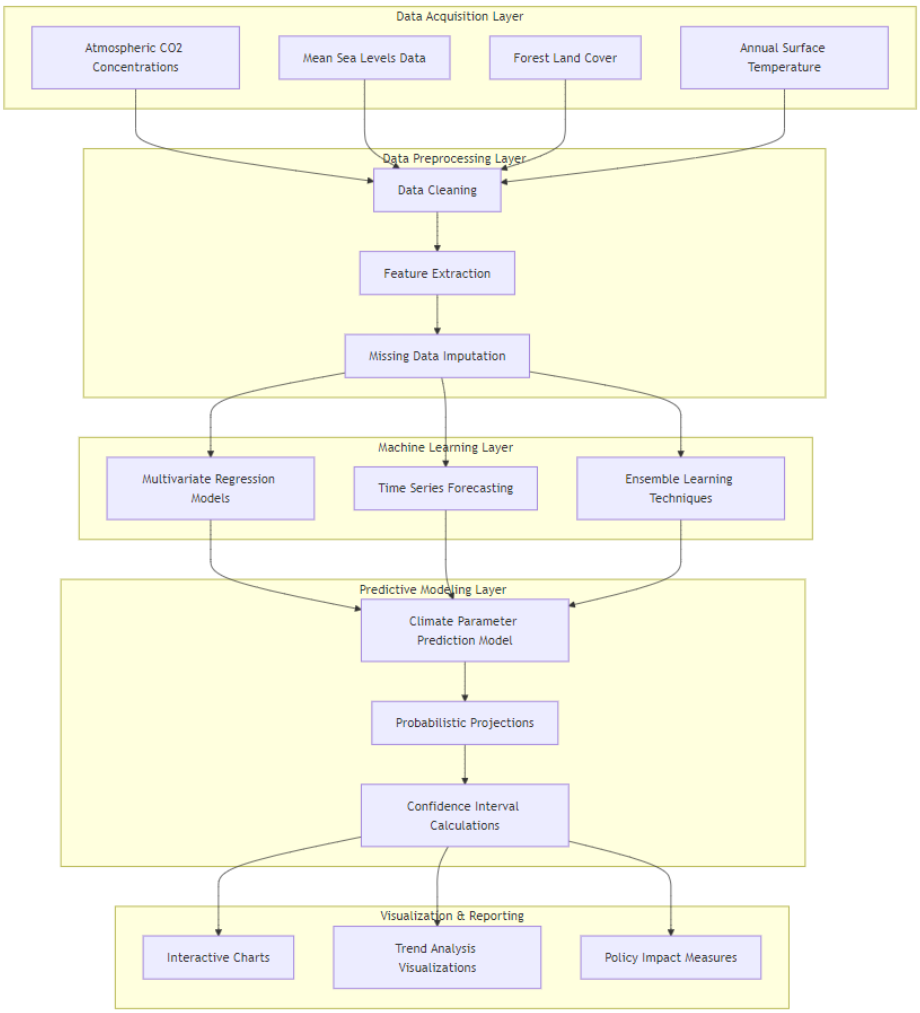


Figure 4.1

1. **Data Collection and Preprocessing:**
    1. Gather historical data on the above indicators from the IMF's Climate Data Portal.
    2. Clean, preprocess, and transform data to ensure consistency and completeness for use in predictive models.
2. **Exploratory Data Analysis (EDA):**
    1. Analyze the trends, patterns, and distributions of the climate indicators over time.
    2. Identify potential outliers, missing values, and other anomalies.
3. **Modeling and Prediction:**
    1. Develop predictive models (such as Regression, ARIMA, LSTM etc.) to forecast future values of each indicator.
    2. Evaluate model performance using appropriate metrics (e.g., RMSE, MAE) and optimize the model for better accuracy.

Research Design:

The research design involves a combination of statistical and machine learning techniques to develop predictive models for key climate change parameters in India. This mixed-method approach allows for robust analysis, leveraging the strengths of both traditional and advanced methodologies.

Rationale for Chosen Methodology:

- **Linear and Multiple Linear Regression**: Chosen for their simplicity and interpretability, providing a baseline for understanding the relationships between climate variables.
- **ARIMA:** Selected for its capability to handle time-series data and capture linear trends and seasonal patterns, making it suitable for ***short-term forecasting.***
- **LSTM:** Employed for its ability to model long-term dependencies and complex, non-linear relationships in sequential data, crucial for accurate ***long-term climate predictions.***

Data Sources, Collection Methods, and Instruments

Data Sources:

- Primary Source: International Monetary Fund **(IMF) Climate Data Platform**, providing comprehensive historical global and regional climate data specific to India.

Collection Methods:

- **Data Extraction**: Downloading relevant datasets from the IMF Climate Data Platform.
- **Data Cleaning:** Removing any inconsistencies, handling missing values, and ensuring data quality.

Instruments Used:

- **Software:** *Python.*
- **Libraries and Frameworks**: Pandas, NumPy, scikit-learn for regression models, statsmodels for ARIMA, and TensorFlow, Keras , sklearn for LSTM models.

Data Analysis Procedures

1. **Exploratory Data Analysis (EDA):**
    - Visualization: Plotting time-series data to identify trends, seasonal patterns, and anomalies.
    - Descriptive Statistics: Calculating relevant statistical information.
2. **Model Development:**
    - Linear and Multiple Linear Regression: Estimating relationships between dependent and independent variables.
    - ARIMA: Selecting appropriate p, d, and q parameters using methods like auto-correlation and partial auto-correlation plots and auto ARIMA.
    - LSTM: Designing and Training the model on historical climate data.
3. **Model Validation:**
    - Performance Metrics: Evaluating models using Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared.
4. **Scenario Analysis:**
    - Developing and analyzing different climate scenarios to predict future states and assess their potential impacts on India's ecosystems, human health, and economy.

Validity and Reliability

Ensuring Validity:

- **Data Quality:** Ensuring that the data from the IMF Climate Data Platform is accurate, consistent, and relevant.
- **Appropriate Models:** Selecting models that are well-suited for the nature of the data and the research objectives

5. TOOLS AND TECHNIQUES

Python Libraries Used

1. Pandas

- **Functionality**:
    - **Data Manipulation**: Provides data structures like DataFrames to manipulate and analyze structured data easily.
    - **Data Cleaning**: Functions for handling missing data, filtering rows/columns, and merging datasets.
    - **Exploratory Data Analysis (EDA)**: Tools to generate descriptive statistics and basic visualizations.
- **Use Case**: Used for loading, cleaning, manipulating, and exploring climate datasets.

2. NumPy

- **Functionality**:
    - **Numerical Operations**: Provides support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.
    - **Linear Algebra**: Functions for performing linear algebra operations, which are essential for statistical analyses.
- **Use Case**: Used for numerical computations and handling array operations efficiently.

3. Matplotlib

- **Functionality**:
    - **Data Visualization**: A comprehensive library for creating static, animated, and interactive visualizations in Python.
    - **Plotting**: Functions to create various types of plots such as line charts, bar charts, histograms, and scatter plots.
- **Use Case**: Used for visualizing climate data and model outputs, helping to identify trends and patterns.

4. Seaborn

- **Functionality**:
    - **Statistical Visualizations**: Built on top of Matplotlib, it provides a high-level interface for drawing attractive and informative statistical graphics.
    - **Enhanced Plots**: Functions for more complex visualizations such as heatmaps, pair plots, and violin plots.
- **Use Case**: Used for creating more aesthetically pleasing and informative statistical plots.

5. Scikit-Learn

- **Functionality**:
    - **Machine Learning**: Tools for predictive data analysis, including classification, regression, clustering, and dimensionality reduction algorithms.
    - **Model Selection**: Functions for hyperparameter tuning and model evaluation.
    - **Preprocessing**: Tools for data preprocessing, including scaling, normalization, and splitting datasets.
- **Use Case**: Used for building and evaluating Linear Regression and other machine learning models.

6. Statsmodels

- **Functionality**:
    - **Statistical Models**: Provides classes and functions for estimating and testing statistical models, including ARIMA and other time-series models.
    - **Statistical Tests**: Tools for performing hypothesis tests and statistical data analysis.
- **Use Case**: Used for implementing ARIMA models and performing advanced statistical analyses.

7. TensorFlow, Keras

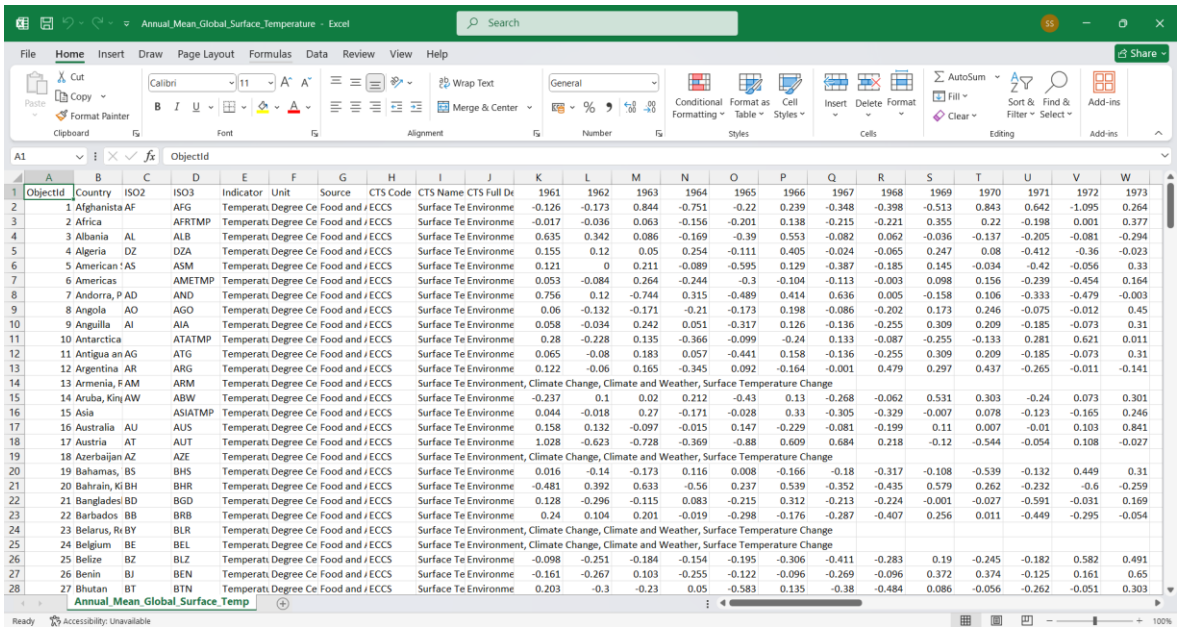
- **Functionality**:
    - **Deep Learning**: TensorFlow is an open-source library for machine learning and deep learning. Keras is a high-level API within TensorFlow for building and training neural networks.
    - **Model Building**: Tools for constructing, training, and evaluating deep learning models, including LSTM networks.
    - **GPU Acceleration**: Support for training models on GPUs for faster computation.
- **Use Case**: Used for building, training, and evaluating LSTM models for long-term climate predictions.

6 IMPLEMENTATIONS

6.1 Mean Global Surface Temperature Dataset

6.1.1. ABOUT THE DATASET:

- **Source of the Dataset : https://climatedata.imf.org/pages/access-data**



ObjectId	Country	ISO2	ISO3	Indicator	Unit	Source	CTS Code	CTS Name	CTS Full De	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971	1972	1973
1	Afghanista	AF	AFG	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	-0.126	-0.173	0.844	-0.751	-0.22	0.239	-0.348	-0.398	-0.513	0.843	0.642	-1.095	0.264
2	Africa		AFRTMP	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	-0.017	-0.036	0.063	-0.156	-0.201	0.138	-0.215	-0.221	0.355	0.22	-0.198	0.001	0.377
3	Albania	AL	ALB	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.635	0.342	0.086	-0.169	-0.39	0.553	-0.082	0.062	-0.036	-0.137	-0.205	-0.081	-0.294
4	Algeria	DZ	DZA	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.155	0.12	0.05	0.254	-0.111	0.405	-0.024	-0.065	0.247	0.08	-0.412	-0.36	-0.023
5	American S	AS	ASM	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.121	0	0.211	-0.089	-0.595	0.129	-0.387	-0.185	0.145	-0.034	-0.42	-0.056	0.33
6	Americas		AMETMP	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.053	-0.084	0.264	-0.244	-0.3	-0.104	-0.113	-0.003	0.098	0.156	-0.239	-0.454	0.164
7	Andorra, P	AD	AND	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.756	0.12	-0.744	0.315	-0.489	0.414	0.636	0.005	-0.158	0.106	-0.333	-0.479	-0.003
8	Angola	AO	AGO	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.06	-0.132	-0.171	-0.21	-0.173	0.198	-0.086	-0.202	0.173	0.246	-0.075	-0.012	0.45
9	Anguilla	AI	AIA	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.058	-0.034	0.242	0.051	-0.317	0.126	-0.136	-0.255	0.309	0.209	-0.185	-0.073	0.31
10	Antarctica		ATATMP	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.28	-0.228	0.135	-0.366	-0.099	-0.24	0.133	-0.087	-0.255	-0.133	0.281	0.621	0.011
11	Antigua an	AG	ATG	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.065	-0.08	0.183	0.057	-0.441	0.158	-0.136	-0.255	0.309	0.209	-0.185	-0.073	0.31
12	Argentina	AR	ARG	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.122	-0.06	0.165	-0.345	0.092	-0.164	-0.001	0.479	0.297	0.437	-0.265	-0.011	-0.141
13	Armenia, F	AM	ARM	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environment, Climate Change, Climate and Weather, Surface Temperature Change													
14	Aruba, Kin	AW	ABW	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	-0.237	0.1	0.02	0.212	-0.43	0.13	-0.268	-0.062	0.531	0.303	-0.24	0.073	0.301
15	Asia		ASIATMP	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.044	-0.018	0.27	-0.171	-0.028	0.33	-0.305	-0.329	-0.007	0.078	-0.123	-0.165	0.246
16	Australia	AU	AUS	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.158	0.132	-0.097	-0.015	0.147	-0.229	-0.081	-0.199	0.11	0.007	-0.01	0.103	0.841
17	Austria	AT	AUT	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	1.028	-0.623	-0.728	-0.369	-0.88	0.609	0.684	0.218	-0.12	-0.544	-0.054	0.108	-0.027
18	Azerbaijan	AZ	AZE	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environment, Climate Change, Climate and Weather, Surface Temperature Change													
19	Bahamas,	BS	BHS	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.016	-0.14	-0.173	0.116	0.008	-0.166	-0.18	-0.317	-0.108	-0.539	-0.132	0.449	0.31
20	Bahrain, Ki	BH	BHR	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	-0.481	0.392	0.633	-0.56	0.237	0.539	-0.352	-0.435	0.579	0.262	-0.232	-0.6	-0.259
21	Banglades	BD	BGD	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.128	-0.296	-0.115	0.083	-0.215	0.312	-0.213	-0.224	-0.001	-0.027	-0.591	-0.031	0.169
22	Barbados	BB	BRB	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.24	0.104	0.201	-0.019	-0.298	-0.176	-0.287	-0.407	0.256	0.011	-0.449	-0.295	-0.054
23	Belarus, Re	BY	BLR	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environment, Climate Change, Climate and Weather, Surface Temperature Change													
24	Belgium	BE	BEL	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environment, Climate Change, Climate and Weather, Surface Temperature Change													
25	Belize	BZ	BLZ	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	-0.098	-0.251	-0.184	-0.154	-0.195	-0.306	-0.411	-0.283	0.19	-0.245	-0.182	0.582	0.491
26	Benin	BJ	BEN	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	-0.161	-0.267	0.103	-0.255	-0.122	-0.096	-0.269	-0.096	0.372	0.374	-0.125	0.161	0.65
27	Bhutan	BT	BTN	Temperatu	Degree Ce	Food and A	ECCS	Surface Te	Environme	0.203	-0.3	-0.23	0.05	-0.583	0.135	-0.38	-0.484	0.086	-0.056	-0.262	-0.051	0.303

Key Columns in the Dataset:

1. **ObjectId**: A unique identifier for each record.
2. **Country**: The name of the country or region.
3. **ISO2**: The 2-letter ISO country code (e.g., "AF" for Afghanistan).
4. **ISO3**: The 3-letter ISO country code (e.g., "AFG" for Afghanistan).
5. **Indicator**: Describes the specific indicator being measured. In this case, it's the "Temperature change with respect to a baseline climatology, corresponding to the period."
6. **Unit**: The measurement unit for the data, which is "Degree Celsius" in this dataset.
7. **Source**: Refers to the source of the data. This dataset comes from the **Food and Agriculture Organization of the United Nations (FAO)**.
8. **CTS Code, CTS Name, CTS Full Descriptor**: Codes and descriptors related to the type of climate or environmental indicator being measured.

9. **Years (1961-2023)**: The temperature change data for each year from 1961 to 2023 for each country or region. These values indicate the change in temperature in comparison to the baseline climatology.

6.1.2. Exploratory Data Analysis

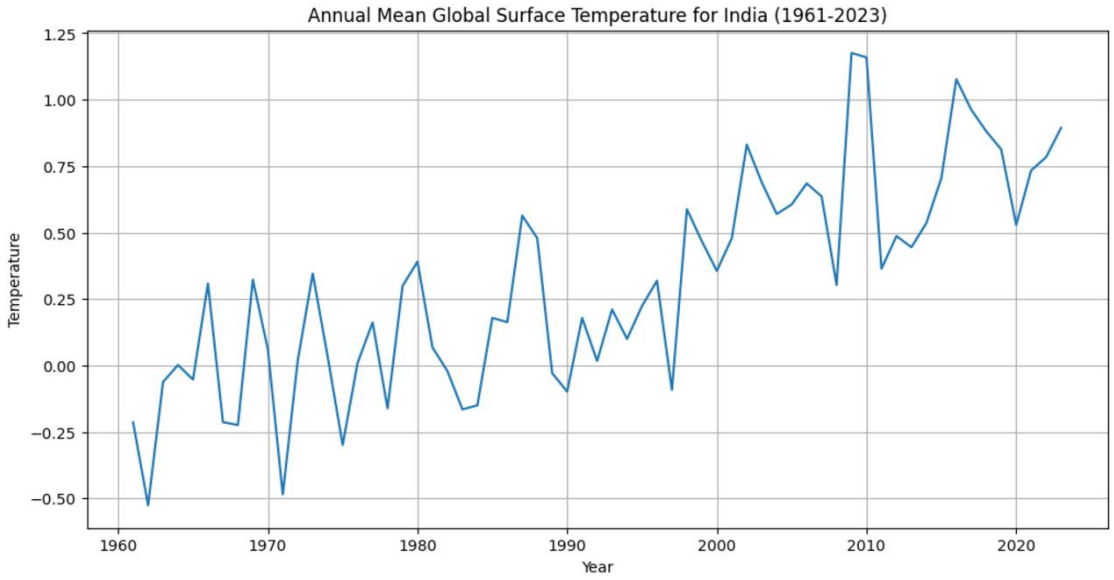


Figure 6.1

Interpretation: Mean Global Surface Temperature seems to be increasing over the period of years plotted above.

6.1.3 Results and Discussion: Mean Global Surface Temperature

LINEAR REGRESSION MODEL:

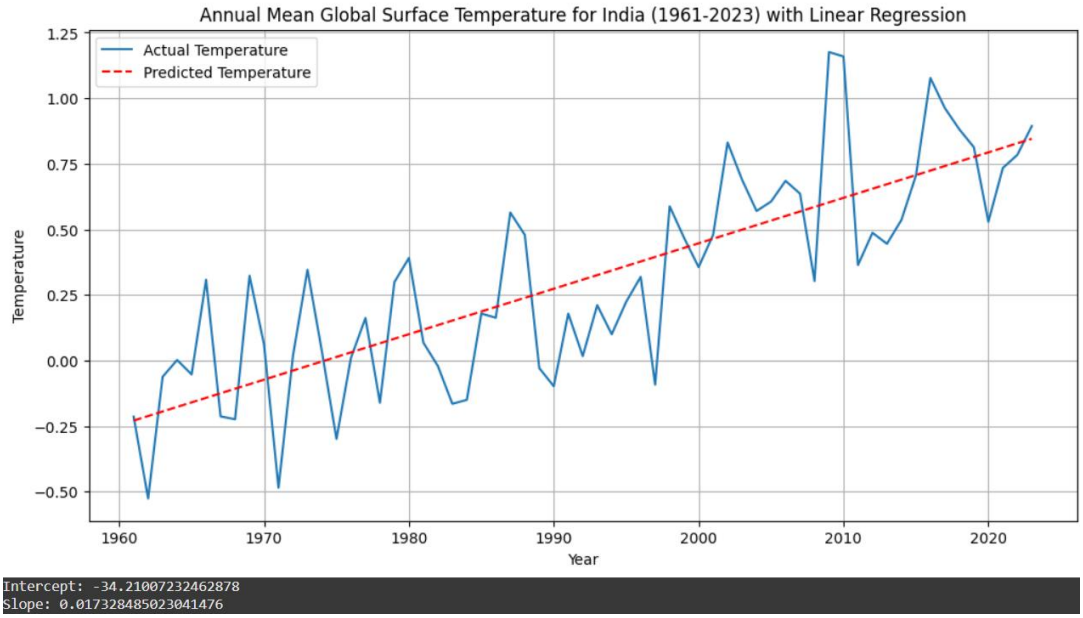


Figure 6.2

PREDICTION USING LINEAR REGRESSION:

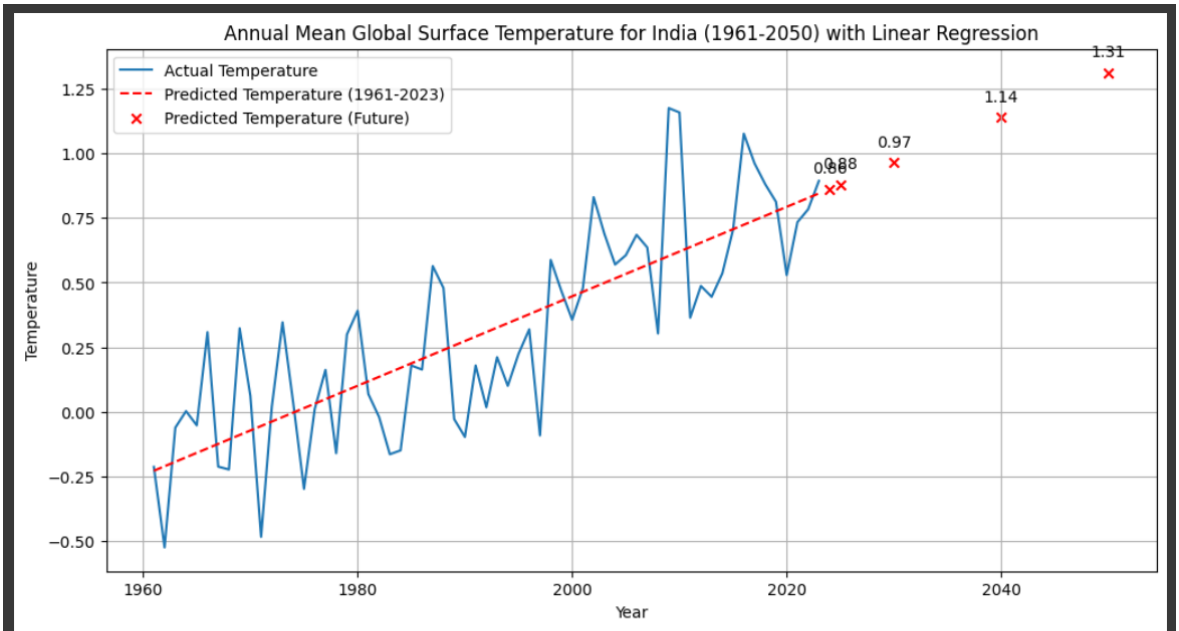


Figure 6.3

EVALUATION METRICS:

```
[ ]  import numpy as np
     from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
     from scipy import stats

     # Calculate Mean Squared Error (MSE)
     mse = mean_squared_error(y, y_pred)
     print(f"Mean Squared Error (MSE): {mse}")

     # Calculate Root Mean Squared Error (RMSE)
     rmse = np.sqrt(mse)
     print(f"Root Mean Squared Error (RMSE): {rmse}")

     # Calculate Mean Absolute Error (MAE)
     mae = mean_absolute_error(y, y_pred)
     print(f"Mean Absolute Error (MAE): {mae}")
```

```
Mean Squared Error (MSE): 0.05746845380453541
Root Mean Squared Error (RMSE): 0.23972578877654238
Mean Absolute Error (MAE): 0.19309536559546114
```

BUILDING ARIMA MODEL:

ACF and PACF Plots: Mean Global Surface Temperature

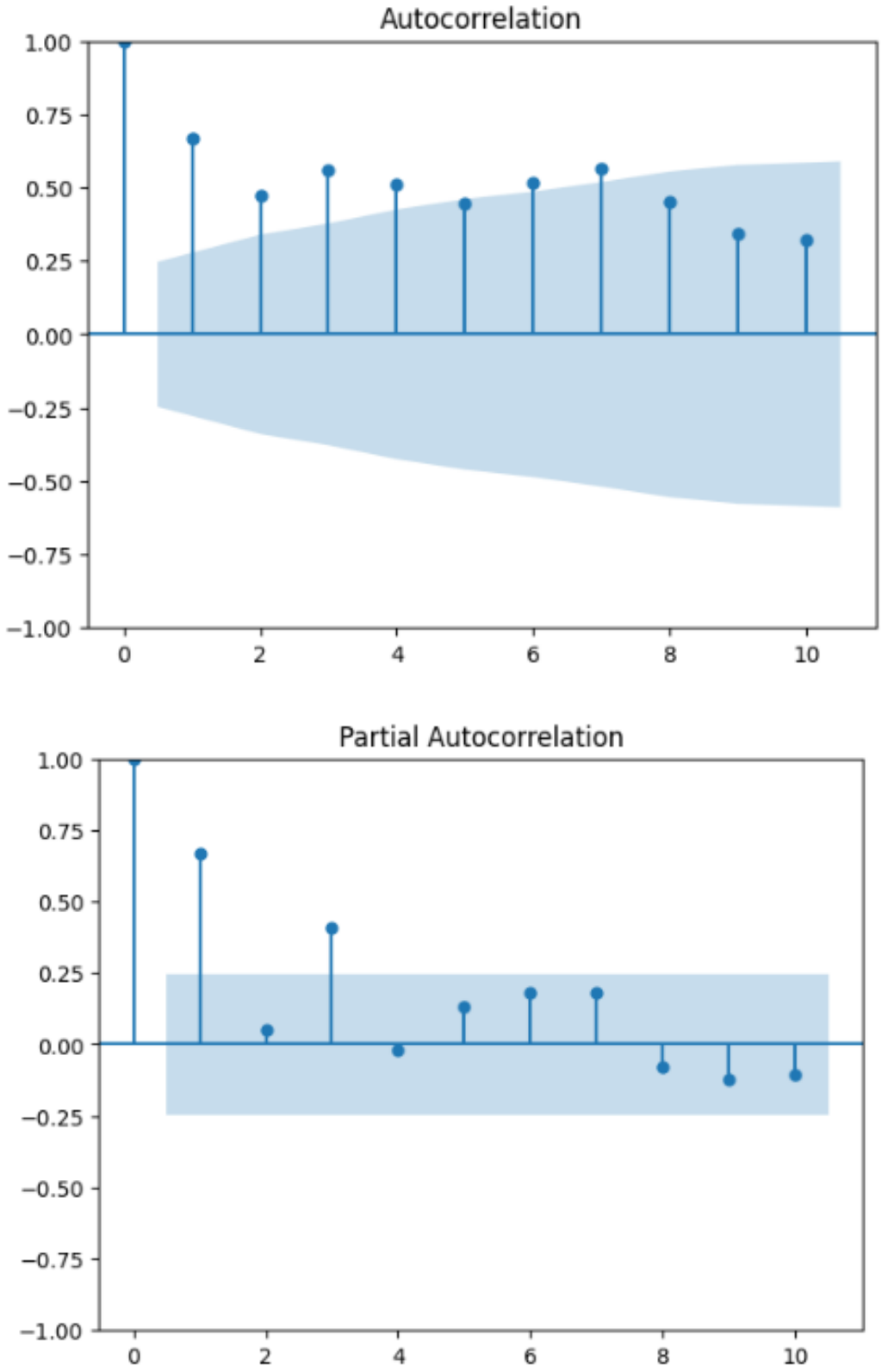


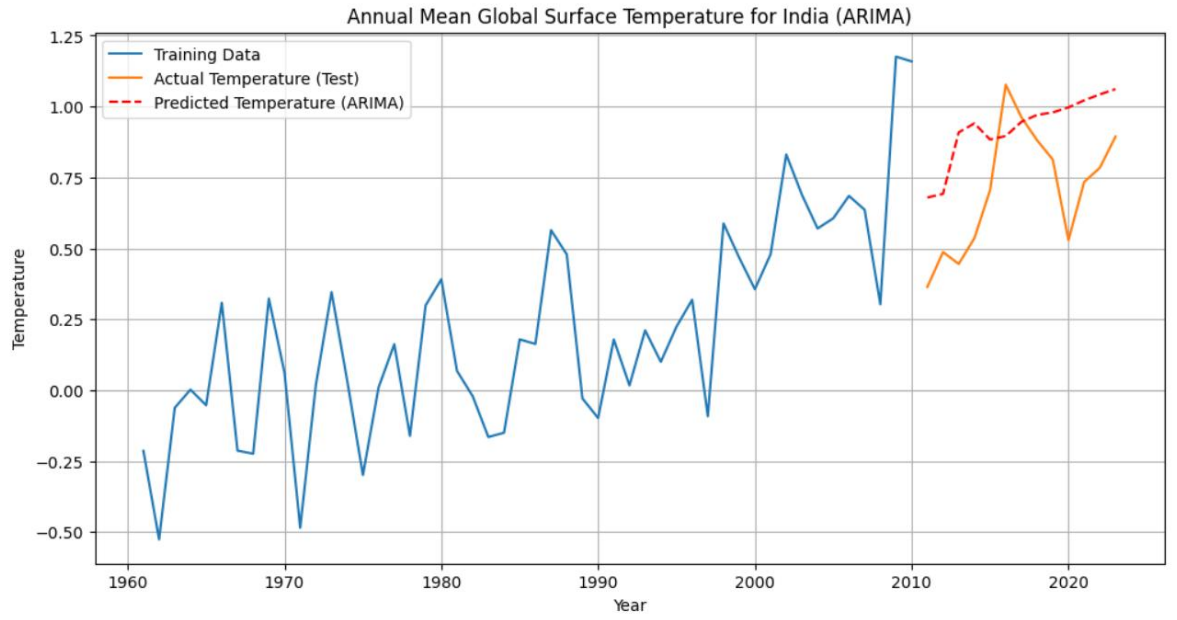
Figure 6.4

Using AUTO ARIMA model to find (p,d,q):

```
Best model:  ARIMA(2,1,1)(0,0,0)[0]
Total fit time: 5.266 seconds
                               SARIMAX Results
==========================================================================================
Dep. Variable:                           y   No. Observations:                       63
Model:                 SARIMAX(2, 1, 1)   Log Likelihood                      -2.386
Date:                  Tue, 10 Dec 2024   AIC                                12.772
Time:                          09:04:14   BIC                                21.281
Sample:                               0   HQIC                               16.113
                                    - 63
Covariance Type:                    opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.1234      0.182      0.677      0.499      -0.234       0.481
ar.L2         -0.3414      0.144     -2.363      0.018      -0.625      -0.058
ma.L1         -0.6848      0.161     -4.267      0.000      -0.999      -0.370
sigma2         0.0622      0.012      5.078      0.000       0.038       0.086
===================================================================================
Ljung-Box (L1) (Q):                   0.08   Jarque-Bera (JB):                 1.14
Prob(Q):                              0.78   Prob(JB):                         0.57
Heteroskedasticity (H):               1.08   Skew:                             0.31
Prob(H) (two-sided):                  0.86   Kurtosis:                         2.78
===================================================================================
```

Table 6.1

Using (p,d,q)=(2,1,1) values to build ARIMA (AutoRegressive Integrated Moving Average)




```
Root Mean Squared Error (RMSE): 0.27978891549344265
Mean Absolute Error (MAE): 0.24630695281713907
```

Figure 6.5

Predicting Values using ARIMA Model:

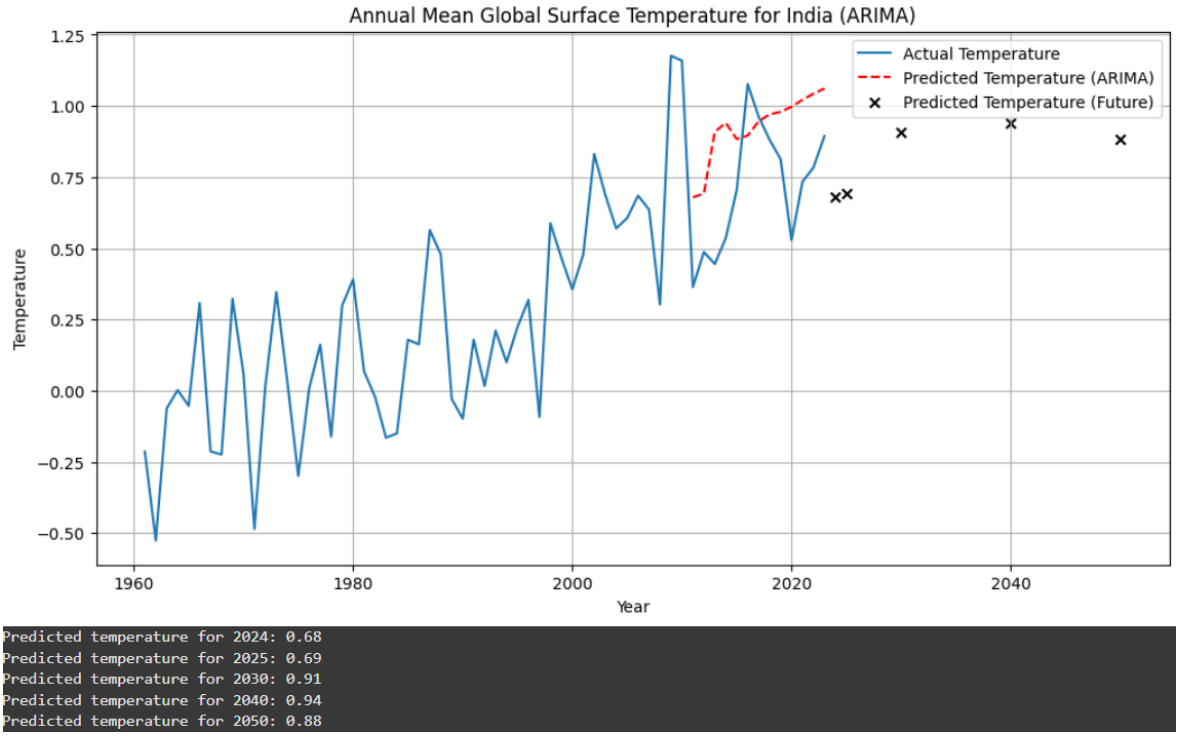


Figure 6.6

Using LSTM (Long Short-Term Memory) Model:

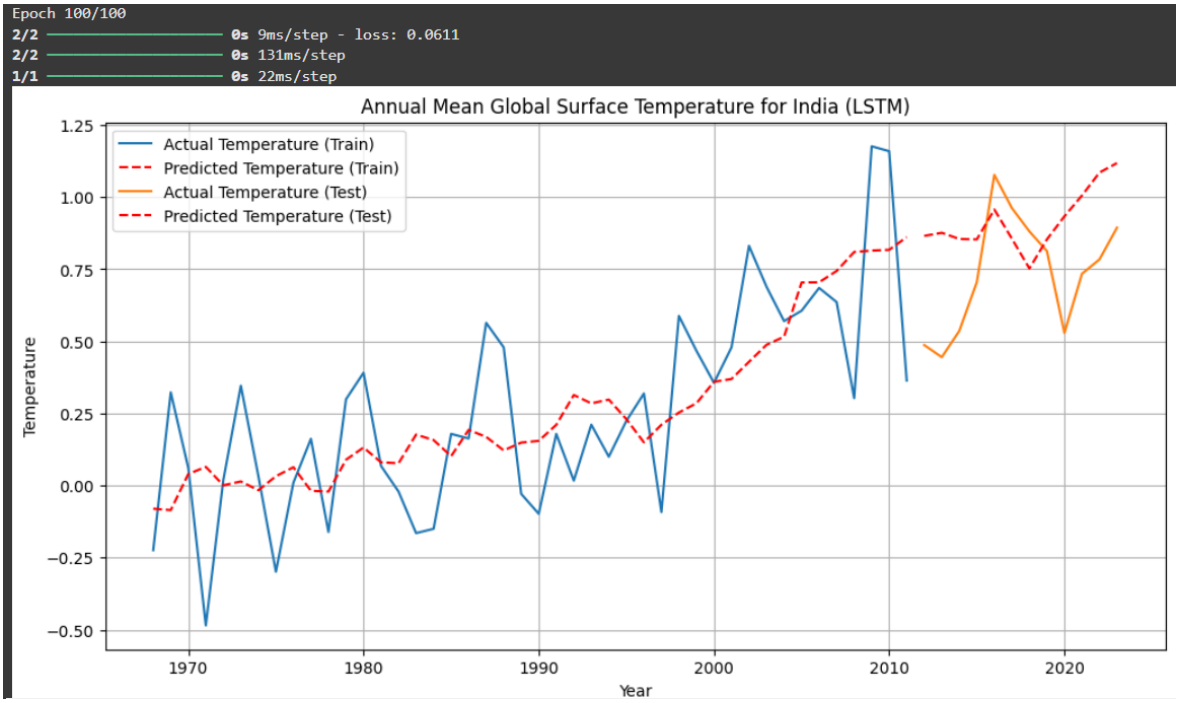
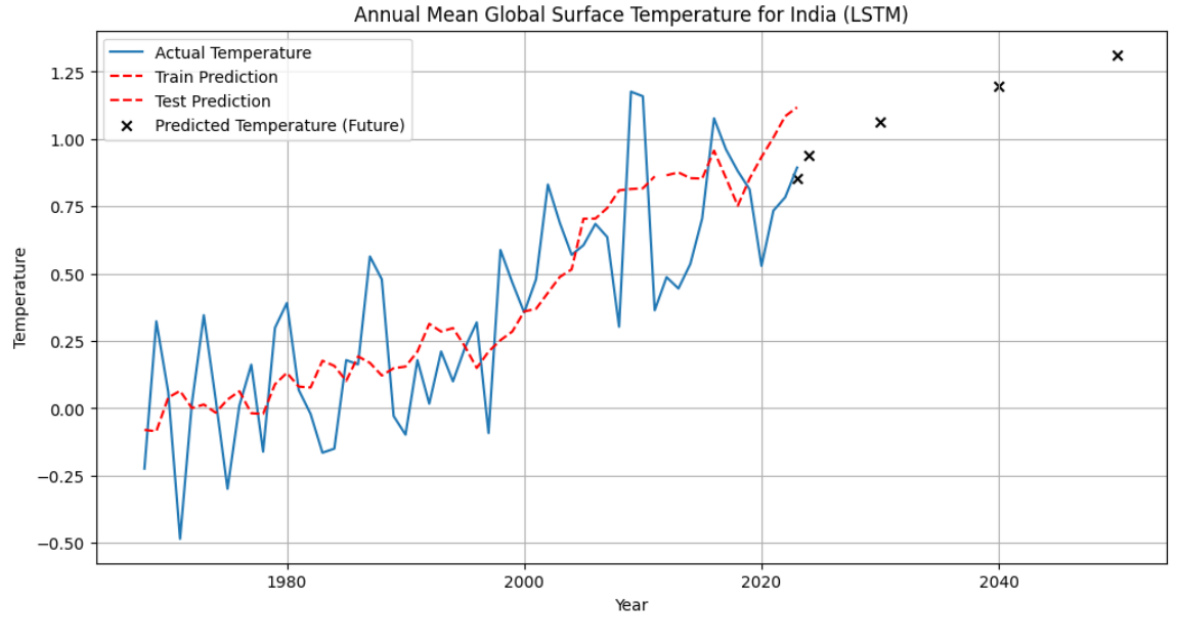


Figure 6.7

Predicting using LSTM (Long Short-Term Memory) Model:




```
Predicted temperature for 2023: 0.85
Predicted temperature for 2024: 0.94
Predicted temperature for 2030: 1.06
Predicted temperature for 2040: 1.20
Predicted temperature for 2050: 1.31
```

Figure 6.8

LSTM Model summary:

```
Train RMSE: 0.2483186996493169
Test RMSE: 0.2160219339208327
Model: "sequential"
```

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 50)	10,400
dense (Dense)	(None, 1)	51

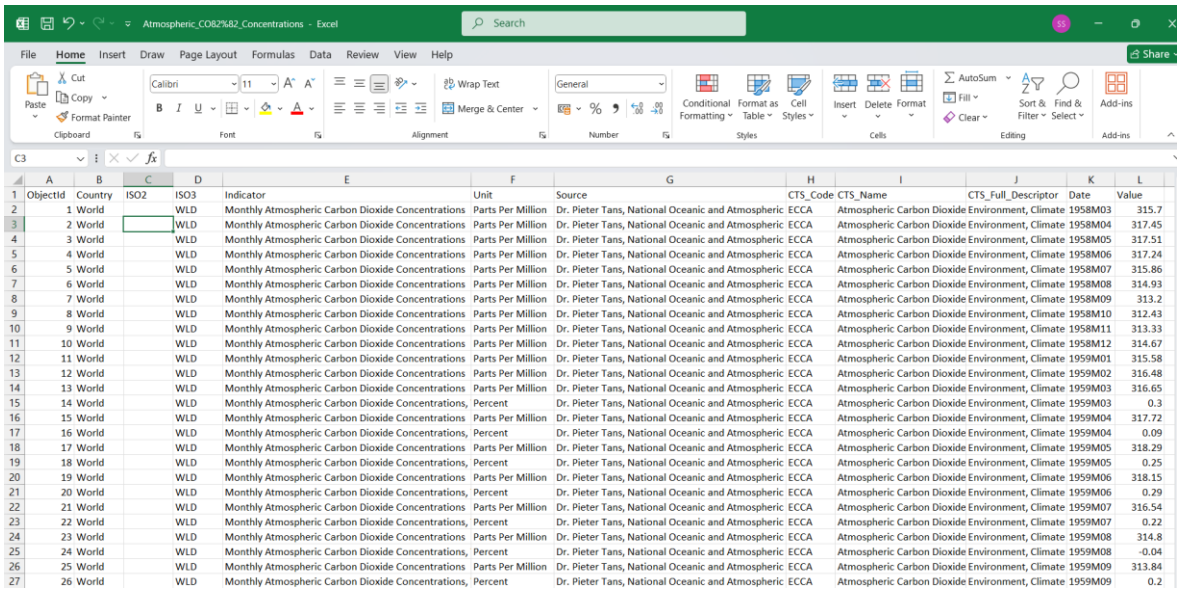
```
 Total params: 31,355 (122.48 KB)
 Trainable params: 10,451 (40.82 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 20,904 (81.66 KB)
None
```

Table 6.2

6.2 Atmospheric Co2 concentration Dataset

6.2.1. ABOUT THE DATASET:

- **Source of the Dataset:** https://climatedata.imf.org/pages/access-data



ObjectId	Country	ISO2	ISO3	Indicator	Unit	Source	CTS_Code	CTS_Name	CTS_Full_Descriptor	Date	Value
1	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1958M03	315.7
2	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1958M04	317.45
3	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1958M05	317.51
4	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1958M06	317.24
5	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1958M07	315.86
6	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1958M08	314.93
7	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1958M09	313.2
8	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1958M10	312.43
9	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1958M11	313.33
10	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1958M12	314.67
11	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M01	315.58
12	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M02	316.48
13	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M03	316.65
14	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Percent	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M03	0.3
15	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M04	317.72
16	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Percent	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M04	0.09
17	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M05	318.29
18	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Percent	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M05	0.25
19	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M06	318.15
20	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Percent	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M06	0.29
21	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M07	316.54
22	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Percent	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M07	0.22
23	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M08	314.8
24	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Percent	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M08	-0.04
25	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Parts Per Million	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M09	313.84
26	World		WLD	Monthly Atmospheric Carbon Dioxide Concentrations	Percent	Dr. Pieter Tans, National Oceanic and Atmospheric	ECCA	Atmospheric Carbon Dioxide	Environment, Climate	1959M09	0.2

Key Columns in the Dataset:

1. **ObjectId**: Unique identifier for each record.
2. **Country**: The geographical entity related to the data, such as "World."
3. **ISO2:** The two-letter ISO country code (e.g., "WLD" for World).
4. **ISO3**: The three-letter ISO country code (e.g., "WLD" for World).
5. **Indicator**: The type of data recorded. In this case, it is "Monthly Atmospheric Carbon Dioxide Concentrations" or "Year on Year Percentage Change."
6. **Unit:** The unit of measurement for the values, such as "Parts Per Million" (PPM) or "Percent."
7. **Source:** The source of the data, which includes references to institutions like NOAA and Scripps Institution of Oceanography.
8. **CTS_Code**: A code referring to the category of the data.
9. **CTS_Name**: The name associated with the data category, like "Atmospheric Carbon Dioxide Concentrations."
10. **CTS_Full_Descriptor**: A more detailed descriptor of the category, often mentioning environmental topics related to climate change and weather.
11. **Date**: The date of the measurement, formatted as YYYYMM (e.g., 1958M03 corresponds to March 1958).

12. **Value**: The actual recorded value for the atmospheric CO2 concentration (in PPM) or the year-on-year percentage change.

6.2.2. Exploratory Data Analysis

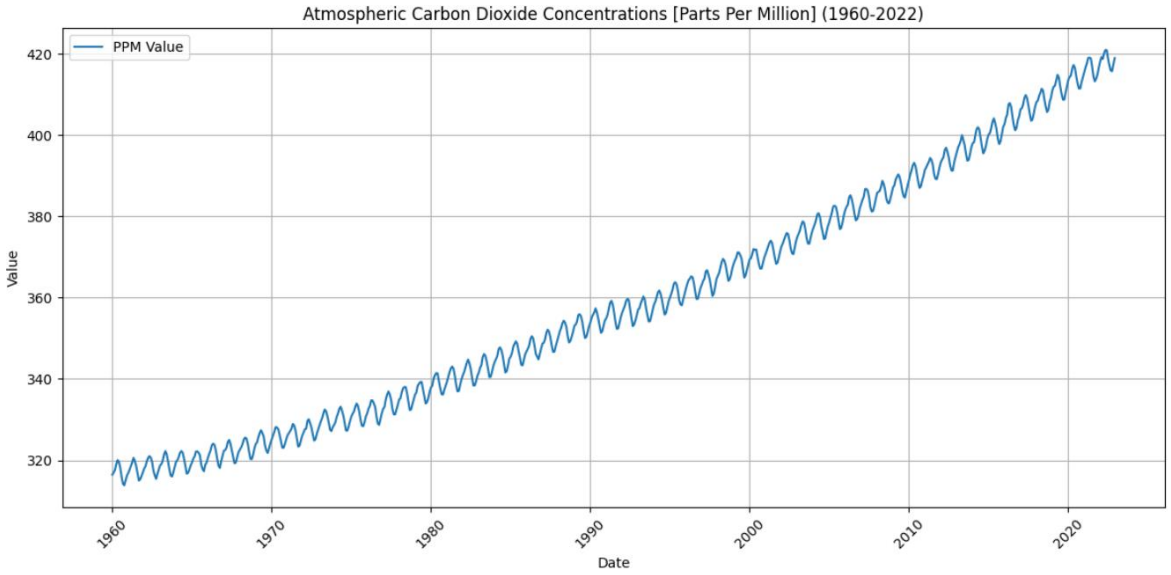


Figure 6.9

Interpretation: Co2 ppm values seems to be increasing over the period of years plotted above.

Results and Discussion: Atmospheric Co2 concentration

LINEAR REGRESSION MODEL:

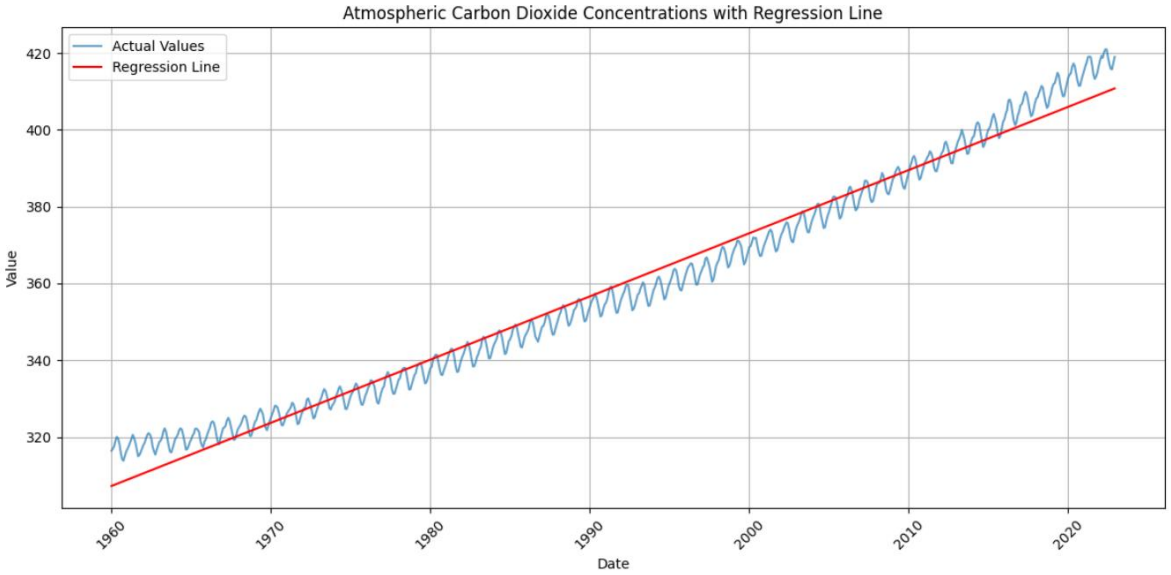


Figure 6.10

PREDICTION USING LINEAR REGRESSION:

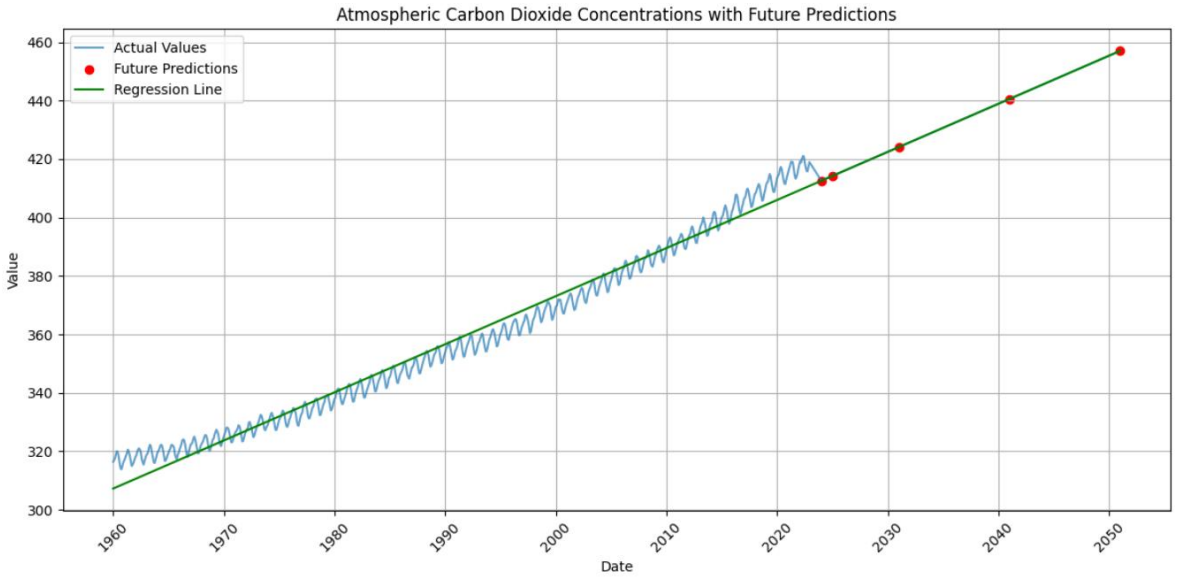


Figure 6.11

```
Future Predictions:
Date: 2023-12-31 00:00:00, Predicted Value: 412.5672139719967
Date: 2024-12-31 00:00:00, Predicted Value: 414.2162673126603
Date: 2030-12-31 00:00:00, Predicted Value: 424.08805930554036
Date: 2040-12-31 00:00:00, Predicted Value: 440.5470534406342
Date: 2050-12-31 00:00:00, Predicted Value: 457.0015419655077
```

EVALUATION METRICS:

```python
# Print the model's coefficients and intercept
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)

# Evaluate the model (example using R-squared)
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)*100
print(f"R-squared: {r2} %")

# You can add other evaluation metrics as needed, such as Mean Squared Error (MSE) or Mean Absolute Error (MAE)
from sklearn.metrics import mean_squared_error, mean_absolute_error
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
```

```
Coefficients: [0.00450561]
Intercept: 307.2485750703215
R-squared: 97.74697051786634 %
Mean Squared Error: 19.107936506686393
Mean Absolute Error: 3.6588814342312617
```

BUILDING ARIMA MODEL:

ACF and PACF Plots: Atmospheric Co2 concentration

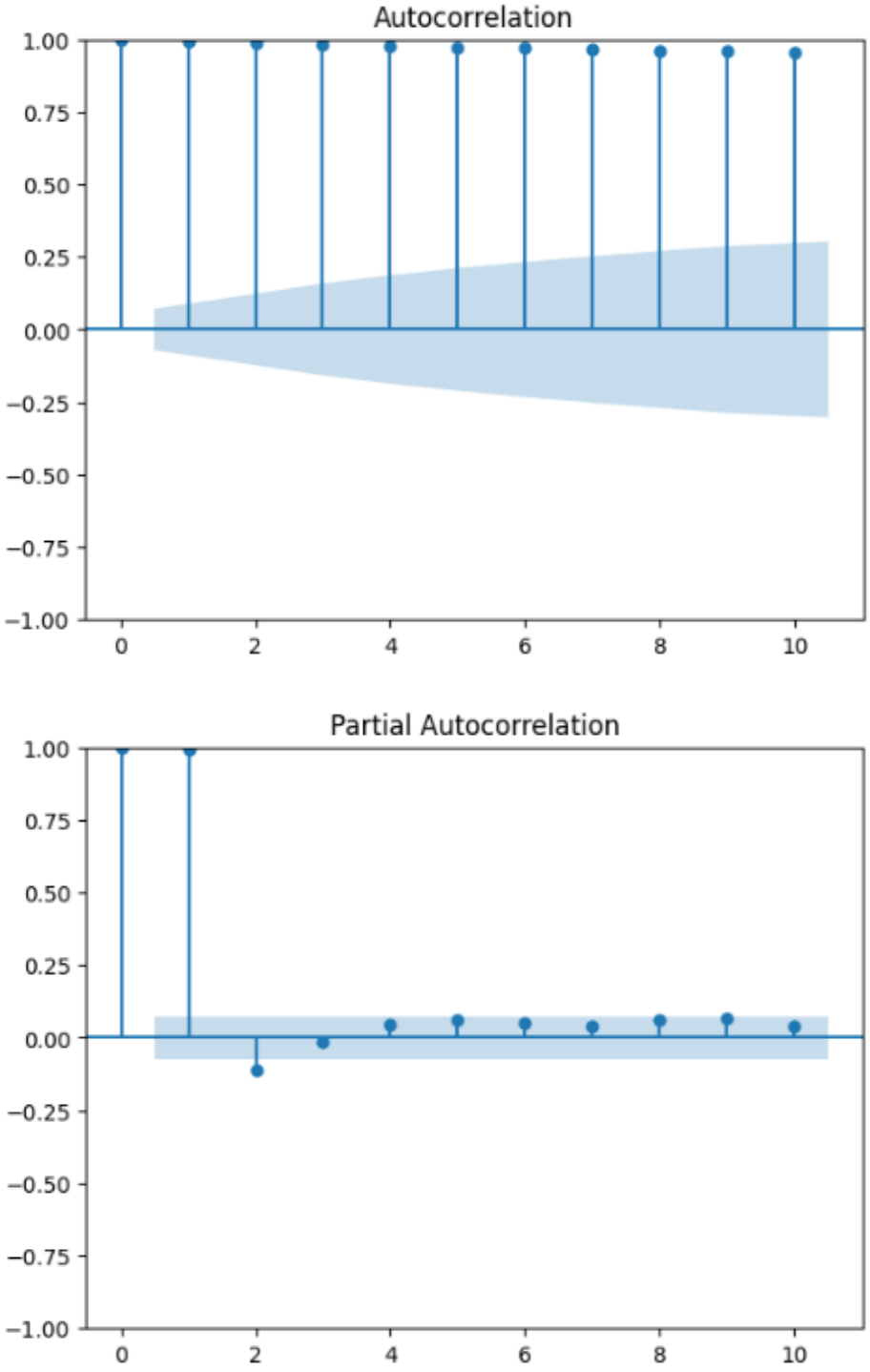


Figure 6.12

Using AUTO ARIMA model to find (p,d,q):

```
Best model:  ARIMA(2,1,1)(0,0,0)[0] intercept
Total fit time: 9.455 seconds
                               SARIMAX Results
==========================================================================================
Dep. Variable:                                y   No. Observations:                  756
Model:                     SARIMAX(2, 1, 1)       Log Likelihood                -749.798
Date:                      Tue, 10 Dec 2024       AIC                           1509.595
Time:                              07:58:45       BIC                           1532.729
Sample:                                   0       HQIC                          1518.506
                                        - 756
Covariance Type:                        opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0412      0.003     11.886      0.000       0.034       0.048
ar.L1          1.5441      0.022     70.329      0.000       1.501       1.587
ar.L2         -0.8465      0.023    -37.316      0.000      -0.891      -0.802
ma.L1         -0.8989      0.018    -49.566      0.000      -0.934      -0.863
sigma2         0.4251      0.022     19.561      0.000       0.382       0.468
===================================================================================
Ljung-Box (L1) (Q):                   0.35   Jarque-Bera (JB):                39.05
Prob(Q):                              0.55   Prob(JB):                         0.00
Heteroskedasticity (H):               1.28   Skew:                             0.52
Prob(H) (two-sided):                  0.05   Kurtosis:                         3.42
===================================================================================
```

Table 6.3

Using (p,d,q)=(2,1,1) values to build ARIMA

```
Future Predictions using ARIMA:
Date: 2023-12-31 00:00:00, Predicted Value: 420.1446884419008
Date: 2024-12-31 00:00:00, Predicted Value: 420.6556712411023
Date: 2030-12-31 00:00:00, Predicted Value: 420.48374825519846
Date: 2040-12-31 00:00:00, Predicted Value: 419.7988454326123
Date: 2050-12-31 00:00:00, Predicted Value: 418.8932374688619
```

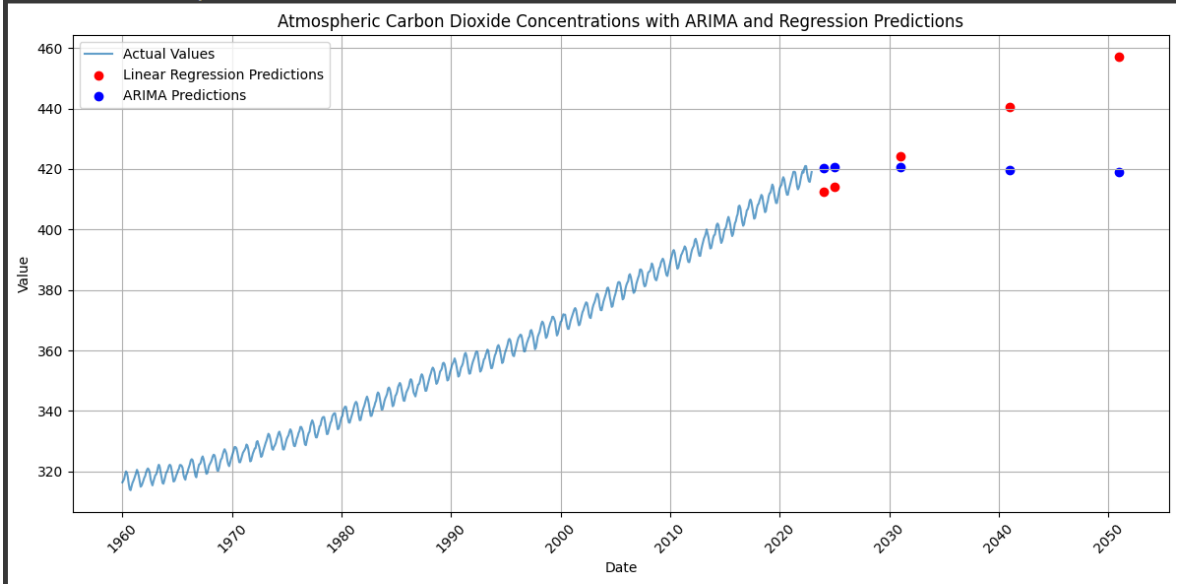


Figure 6.13

Using LSTM (Long Short-Term Memory) Model:

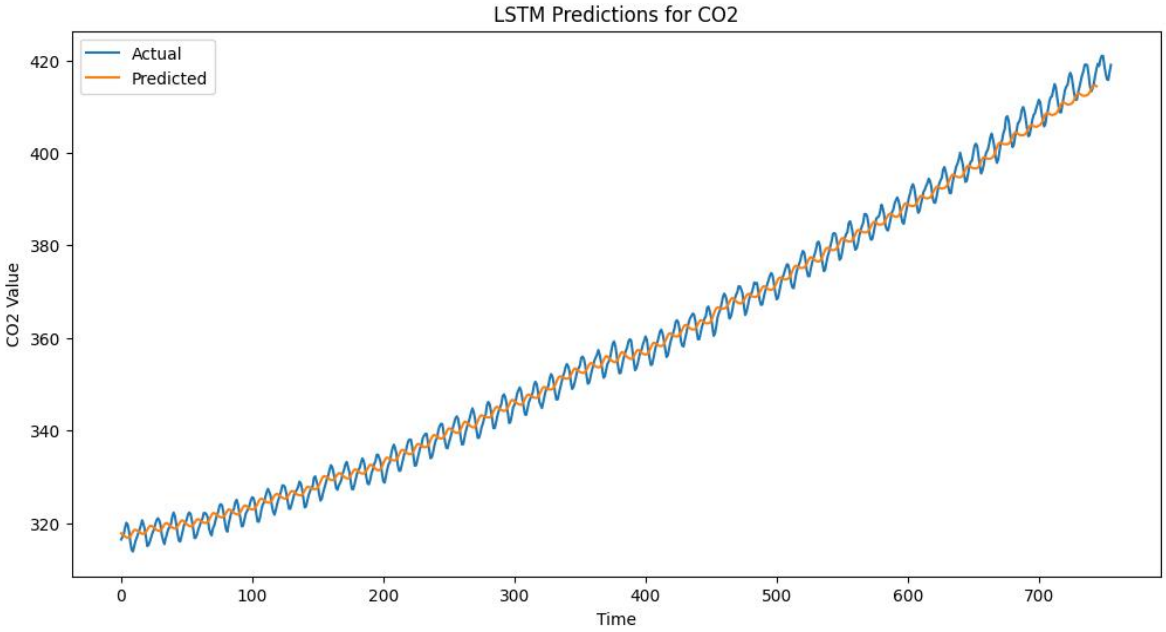


Figure 6.14

Predicting using LSTM (Long Short-Term Memory) Model:

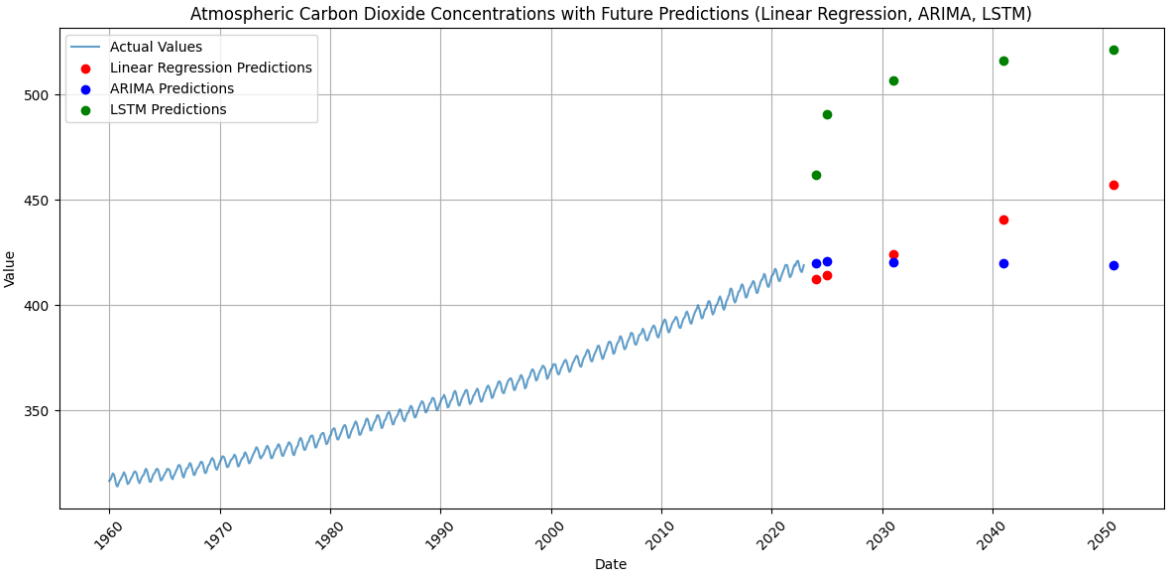


Figure 6.15

LSTM Model summary:

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 10, 50)	10,400
lstm_1 (LSTM)	(None, 50)	20,200
dense (Dense)	(None, 1)	51

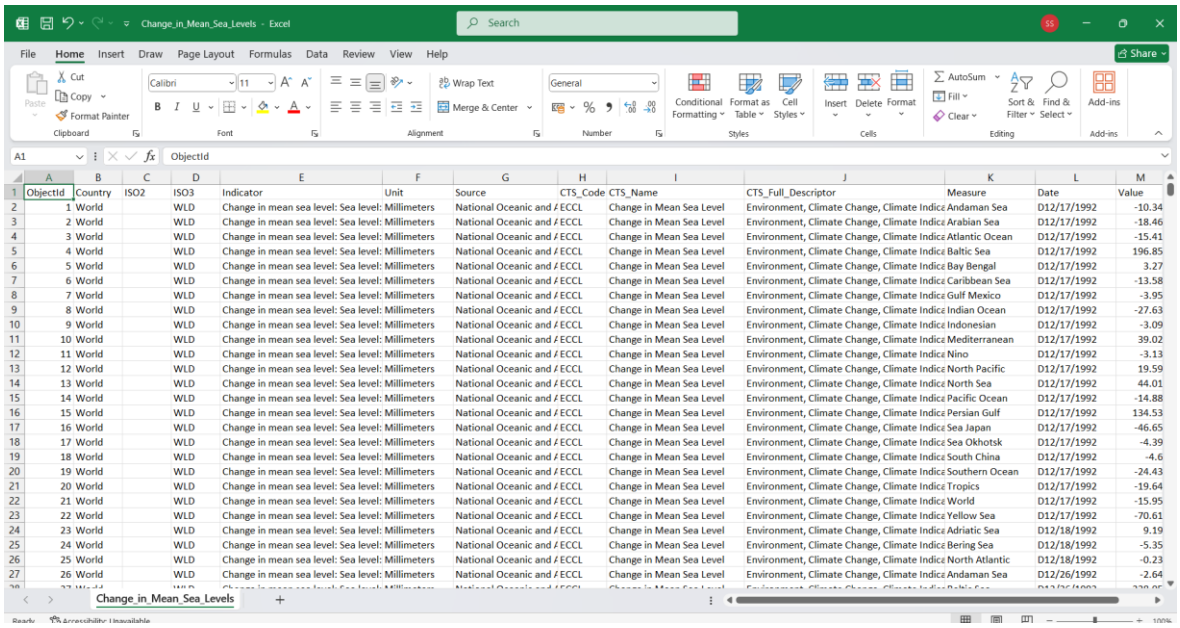
Total params: 91,955 (359.20 KB)
Trainable params: 30,651 (119.73 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 61,304 (239.47 KB)

Table 6.4

6.3 Change in Mean Sea Levels Dataset

6.3.1. ABOUT THE DATASET:

- **Source of the Dataset:** https://climatedata.imf.org/pages/access-data



ObjectId	Country	ISO2	ISO3	Indicator	Unit	Source	CTS_Code	CTS_Name	CTS_Full_Descriptor	Measure	Date	Value
1	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Andaman Sea	D12/17/1992	-10.34
2	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Arabian Sea	D12/17/1992	-18.46
3	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Atlantic Ocean	D12/17/1992	-15.41
4	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Baltic Sea	D12/17/1992	196.85
5	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Bay Bengal	D12/17/1992	3.27
6	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Caribbean Sea	D12/17/1992	-13.58
7	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Gulf Mexico	D12/17/1992	-3.95
8	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Indian Ocean	D12/17/1992	-27.63
9	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Indonesian	D12/17/1992	-3.09
10	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Mediterranean	D12/17/1992	39.02
11	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Nino	D12/17/1992	-3.13
12	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	North Pacific	D12/17/1992	19.59
13	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	North Sea	D12/17/1992	44.01
14	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Pacific Ocean	D12/17/1992	-14.88
15	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Persian Gulf	D12/17/1992	134.53
16	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Sea Japan	D12/17/1992	-46.65
17	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Sea Okhotsk	D12/17/1992	-4.39
18	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	South China	D12/17/1992	-4.6
19	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Southern Ocean	D12/17/1992	-24.43
20	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Tropics	D12/17/1992	-19.64
21	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	World	D12/17/1992	-15.95
22	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Yellow Sea	D12/17/1992	-70.61
23	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Adriatic Sea	D12/18/1992	9.19
24	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Bering Sea	D12/18/1992	-5.35
25	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	North Atlantic	D12/18/1992	-0.23
26	World		WLD	Change in mean sea level: Sea level:	Millimeters	National Oceanic and A	ECCL	Change in Mean Sea Level	Environment, Climate Change, Climate Indica	Andaman Sea	D12/26/1992	-2.64

Key Columns in the Dataset:

1. **ObjectId:** A unique identifier for each row of data.
2. **Country**: The region or country, though all entries are marked "World".
3. **ISO2 and ISO3**: The two-letter and three-letter codes for the region (e.g., "WLD" stands for World).
4. **Indicator:** Describes the type of data (in this case, "Change in mean sea level").
5. **Unit:** Measurement unit (**Millimeters**).
6. **Source**: The source of the data, in this case, NOAA and the TOPEX/Poseidon satellite mission.
7. **CTS_Code**: Code representing the context of the data (e.g., "ECCL").
8. **CTS_Name**: The name related to the environmental context ("Change in Mean Sea Level").
9. **CTS_Full_Descriptor**: Detailed description of the environmental context.
10. **Measure**: The specific location or region the sea level data is related to (e.g., "Andaman Sea", "Baltic Sea").
11. **Date:** The date of the observation.
12. **Value**: The sea level change in millimeters for that specific region on that specific date.

6.3.2. Exploratory Data Analysis

Plotting change in mean sea level for all water bodies:

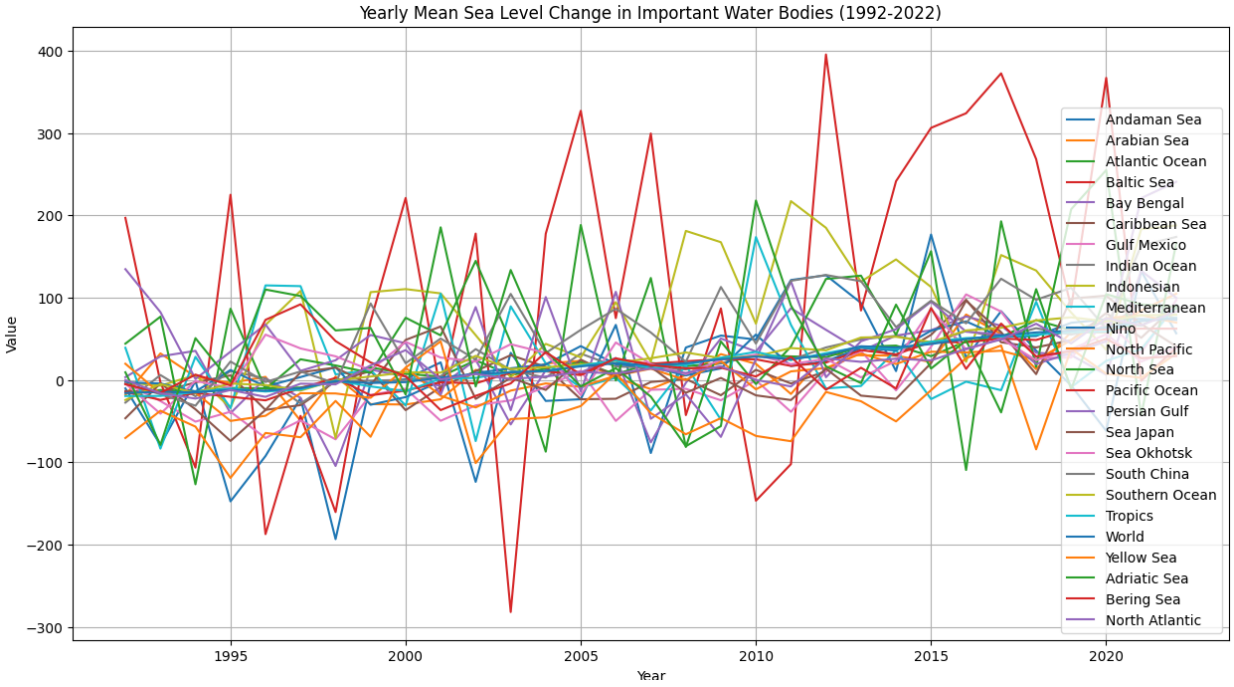


Figure 6.16

Large Water bodies Surrounding India:

Figure 6.17

Mean sea level values for Indian Ocean, Bay of Bengal, Arabian Sea seems to be increasing over the period of years plotted.

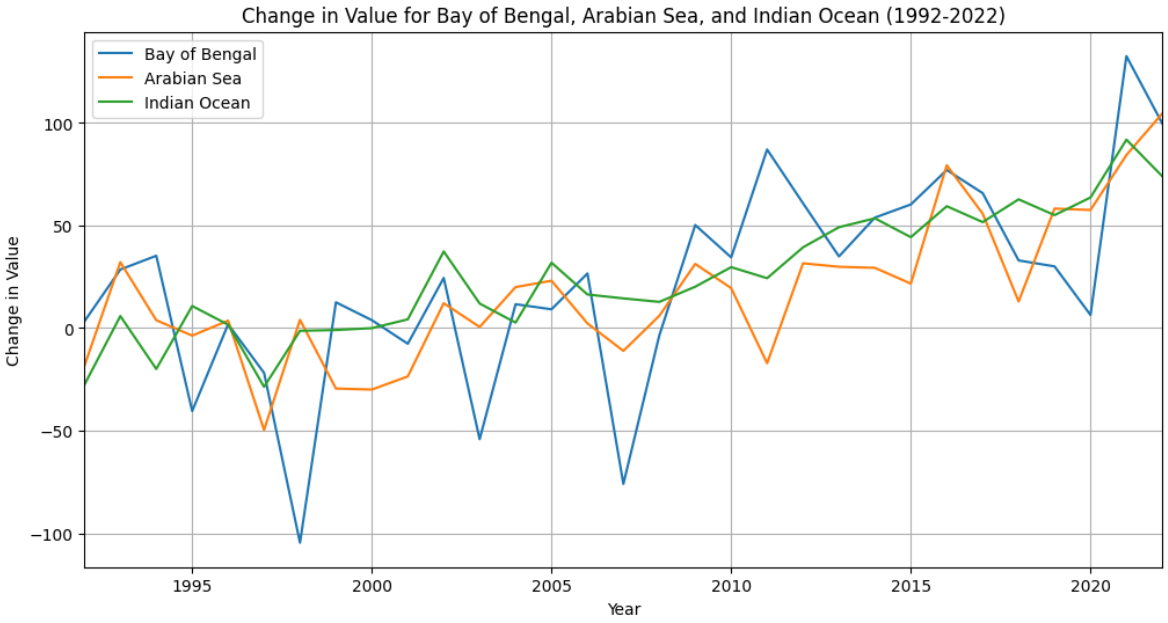


Figure 6.18

Results and Discussion: Change in Mean Sea Levels

LINEAR REGRESSION MODEL:

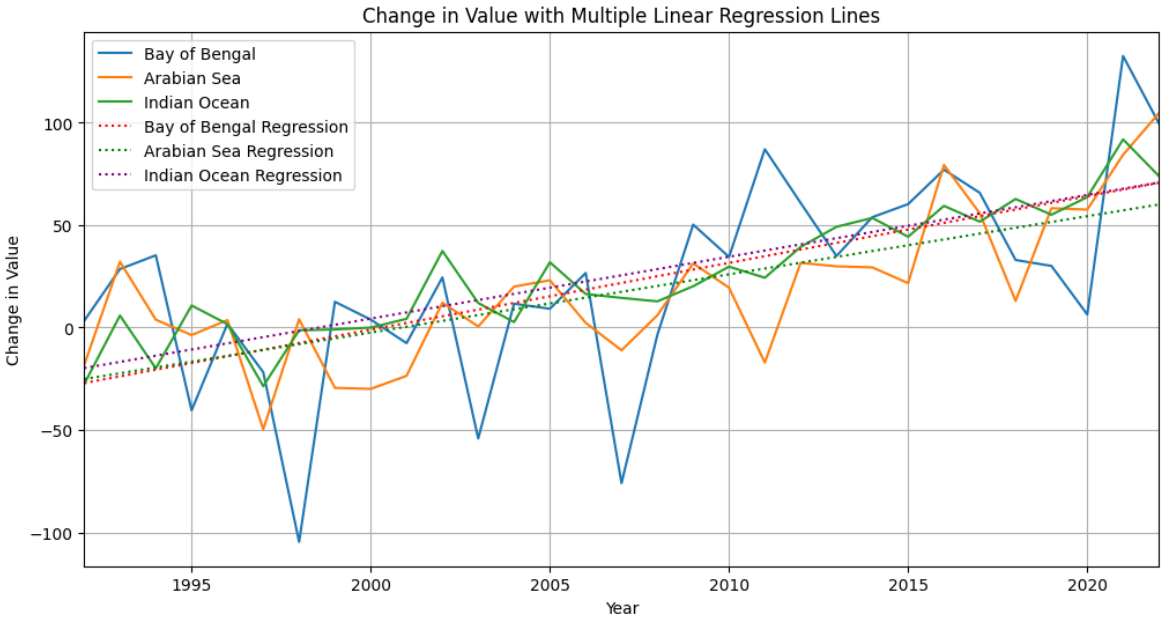


Figure 6.19

PREDICTION USING LINEAR REGRESSION:

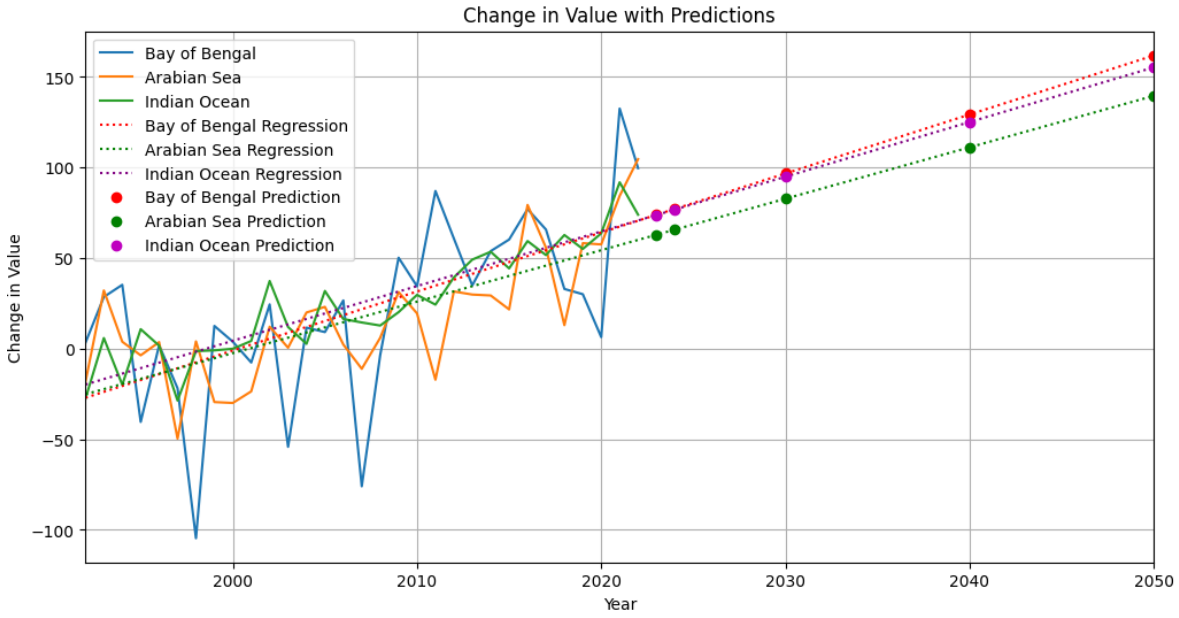


Figure 6.20

Values predicted over time:

```
Predictions for future years:
Year: 2023
  Bay of Bengal: 73.92012903225714
  Arabian Sea: 62.905096774193225
  Indian Ocean: 73.81703225806359
Year: 2024
  Bay of Bengal: 77.17846370967618
  Arabian Sea: 65.74589919354821
  Indian Ocean: 76.83642338709615
Year: 2030
  Bay of Bengal: 96.72847177419226
  Arabian Sea: 82.79071370967722
  Indian Ocean: 94.95277016128966
Year: 2040
  Bay of Bengal: 129.31181854838542
  Arabian Sea: 111.19873790322526
  Indian Ocean: 125.14668145161158
Year: 2050
  Bay of Bengal: 161.89516532257858
  Arabian Sea: 139.6067620967733
  Indian Ocean: 155.3405927419335
```

EVALUATION METRICS:

```
Summary for Indian Ocean:
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.837
Model:                            OLS   Adj. R-squared:                  0.831
Method:                 Least Squares   F-statistic:                     149.0
Date:                Thu, 12 Dec 2024   Prob (F-statistic):           5.97e-13
Time:                        07:19:58   Log-Likelihood:                -120.80
No. Observations:                  31   AIC:                             245.6
Df Residuals:                      29   BIC:                             248.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -6034.4112    496.438    -12.155      0.000   -7049.741   -5019.081
x1             3.0194      0.247     12.207      0.000       2.514       3.525
==============================================================================
Omnibus:                        2.611   Durbin-Watson:                   2.123
Prob(Omnibus):                  0.271   Jarque-Bera (JB):                1.884
Skew:                           0.604   Prob(JB):                        0.390
Kurtosis:                       3.006   Cond. No.                     4.50e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.5e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
Mean Squared Error (MSE): 141.9432567104577
Root Mean Squared Error (RMSE): 11.913994154373993
Mean Absolute Error (MAE): 9.27123803329857
--------------------
```

Table 6.5

```
Summary for Arabian Sea:
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.537
Model:                            OLS   Adj. R-squared:                  0.521
Method:                 Least Squares   F-statistic:                     33.65
Date:                Thu, 12 Dec 2024   Prob (F-statistic):           2.75e-06
Time:                        07:19:58   Log-Likelihood:                -141.97
No. Observations:                  31   AIC:                             287.9
Df Residuals:                      29   BIC:                             290.8
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -5684.0382    982.836     -5.783      0.000   -7694.163   -3673.913
x1             2.8408      0.490      5.801      0.000       1.839       3.842
==============================================================================
Omnibus:                        0.162   Durbin-Watson:                   1.511
Prob(Omnibus):                  0.922   Jarque-Bera (JB):                0.228
Skew:                           0.150   Prob(JB):                        0.892
Kurtosis:                       2.706   Cond. No.                     4.50e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.5e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
Mean Squared Error (MSE): 556.3479337135805
Root Mean Squared Error (RMSE): 23.587028929341237
Mean Absolute Error (MAE): 18.99426404786677
--------------------
```

Table 6.6

```
Summary for Bay of Bengal:
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.358
Model:                            OLS   Adj. R-squared:                  0.336
Method:                 Least Squares   F-statistic:                     16.20
Date:                Thu, 12 Dec 2024   Prob (F-statistic):           0.000374
Time:                        07:19:58   Log-Likelihood:                -157.55
No. Observations:                  31   AIC:                             319.1
Df Residuals:                      29   BIC:                             322.0
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -6517.6909   1624.670     -4.012      0.000   -9840.515   -3194.867
x1             3.2583      0.809      4.025      0.000       1.603       4.914
==============================================================================
Omnibus:                        5.479   Durbin-Watson:                   1.774
Prob(Omnibus):                  0.065   Jarque-Bera (JB):                3.951
Skew:                          -0.829   Prob(JB):                        0.139
Kurtosis:                       3.556   Cond. No.                     4.50e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.5e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
Mean Squared Error (MSE): 1520.250187085066
Root Mean Squared Error (RMSE): 38.99038582888179
Mean Absolute Error (MAE): 29.449229708636853
--------------------
```

Table 6.7

ACF and PACF Plots: Change in Mean Sea Levels

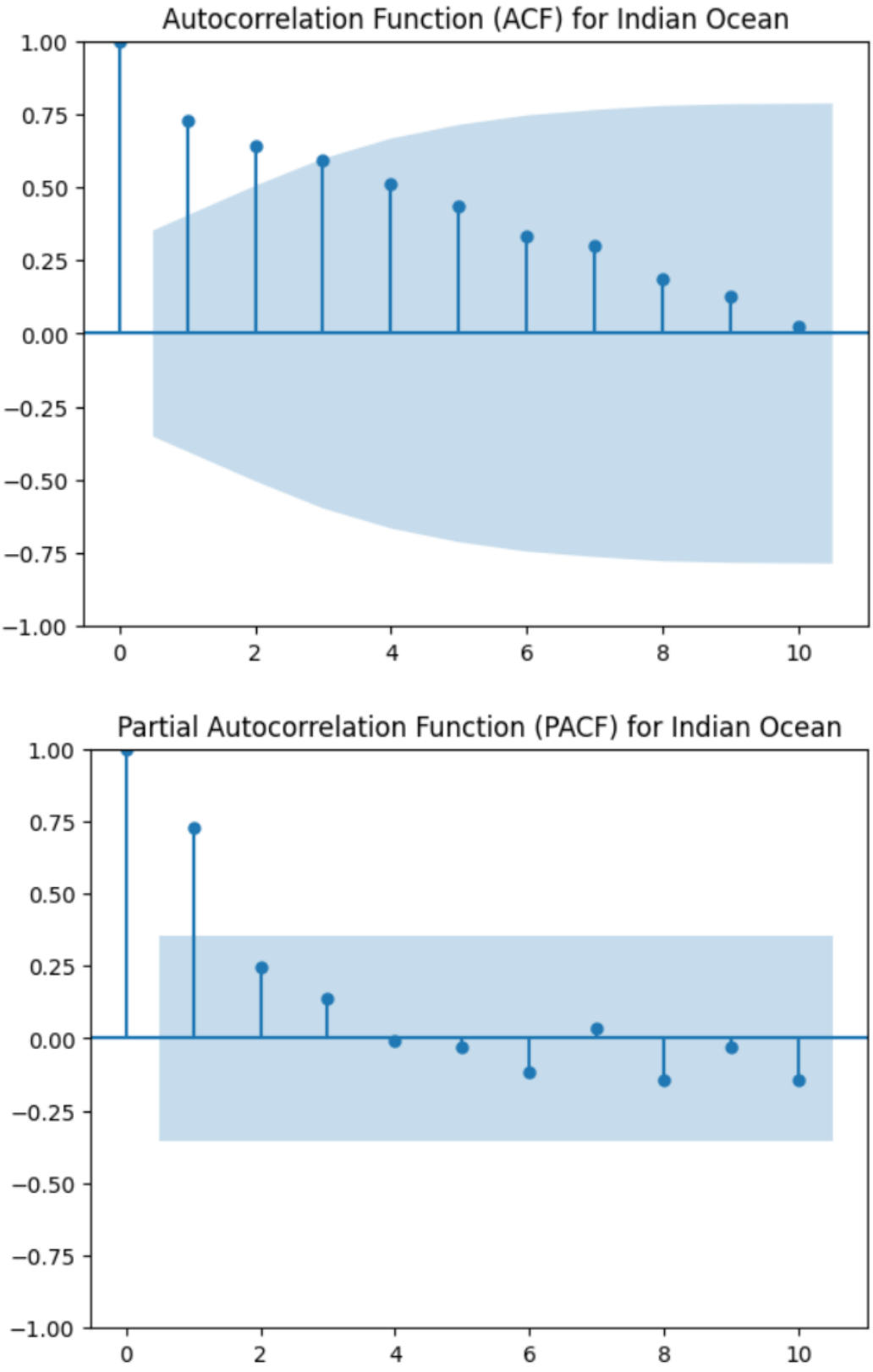


Figure 6.21

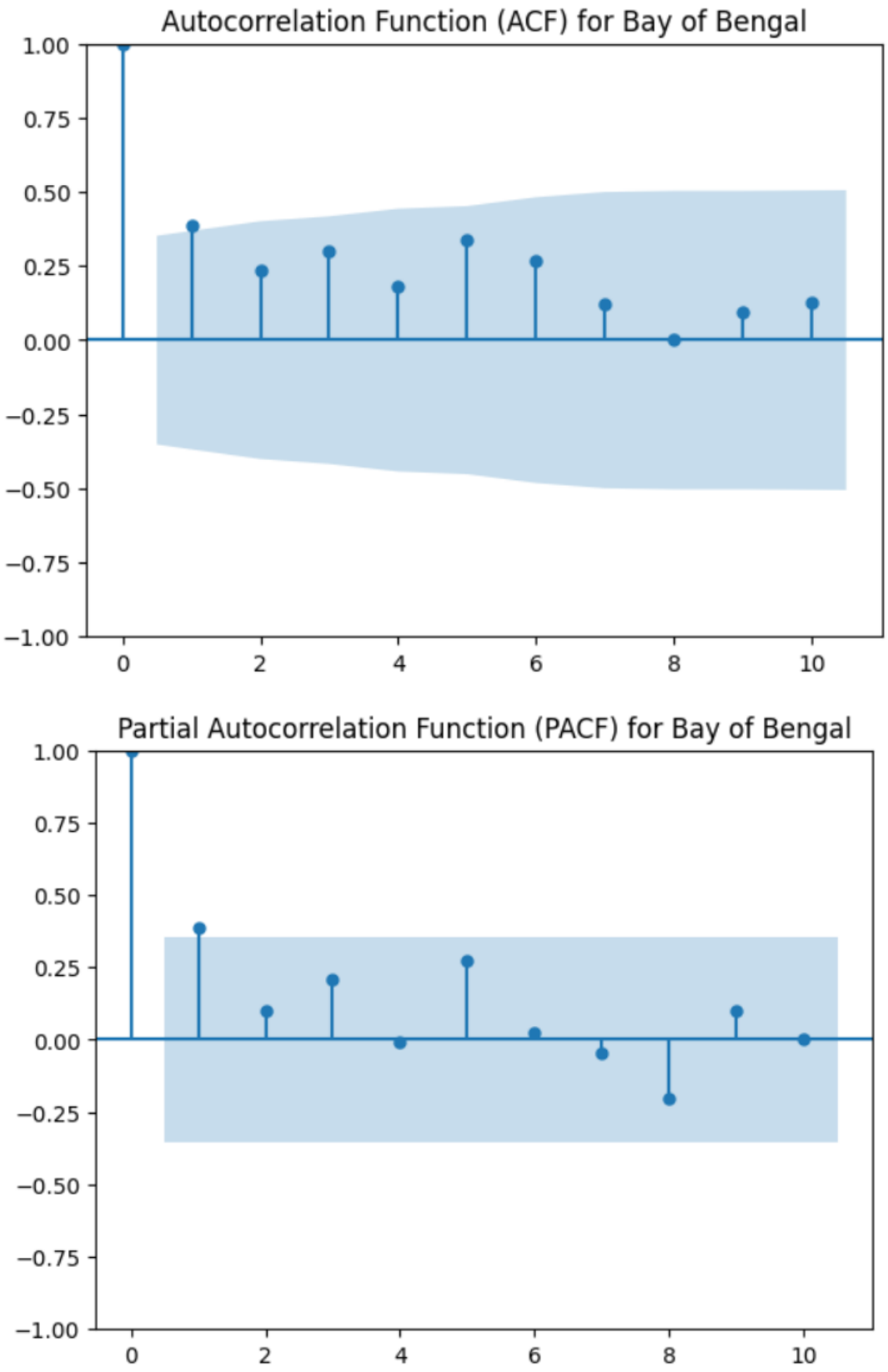
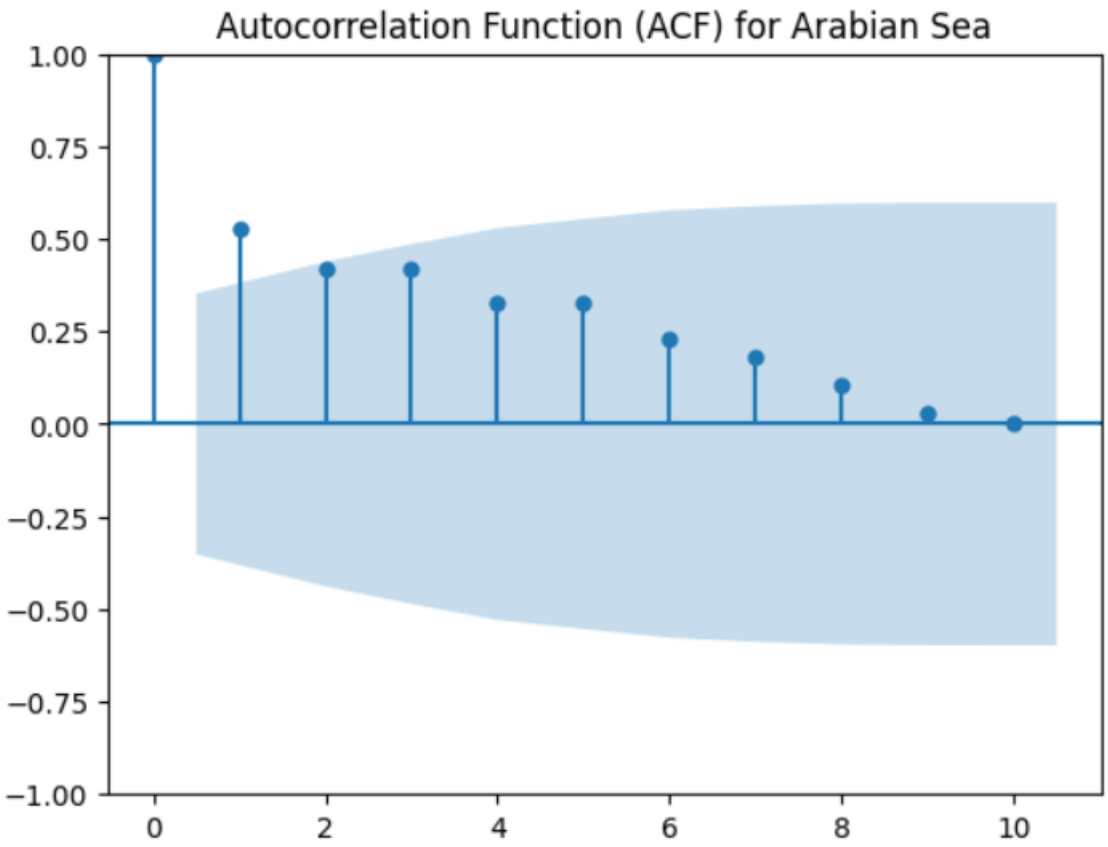

Autocorrelation Function (ACF) for Bay of Bengal
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10
Partial Autocorrelation Function (PACF) for Bay of Bengal
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10

Figure 6.22


Autocorrelation Function (ACF) for Arabian Sea
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10

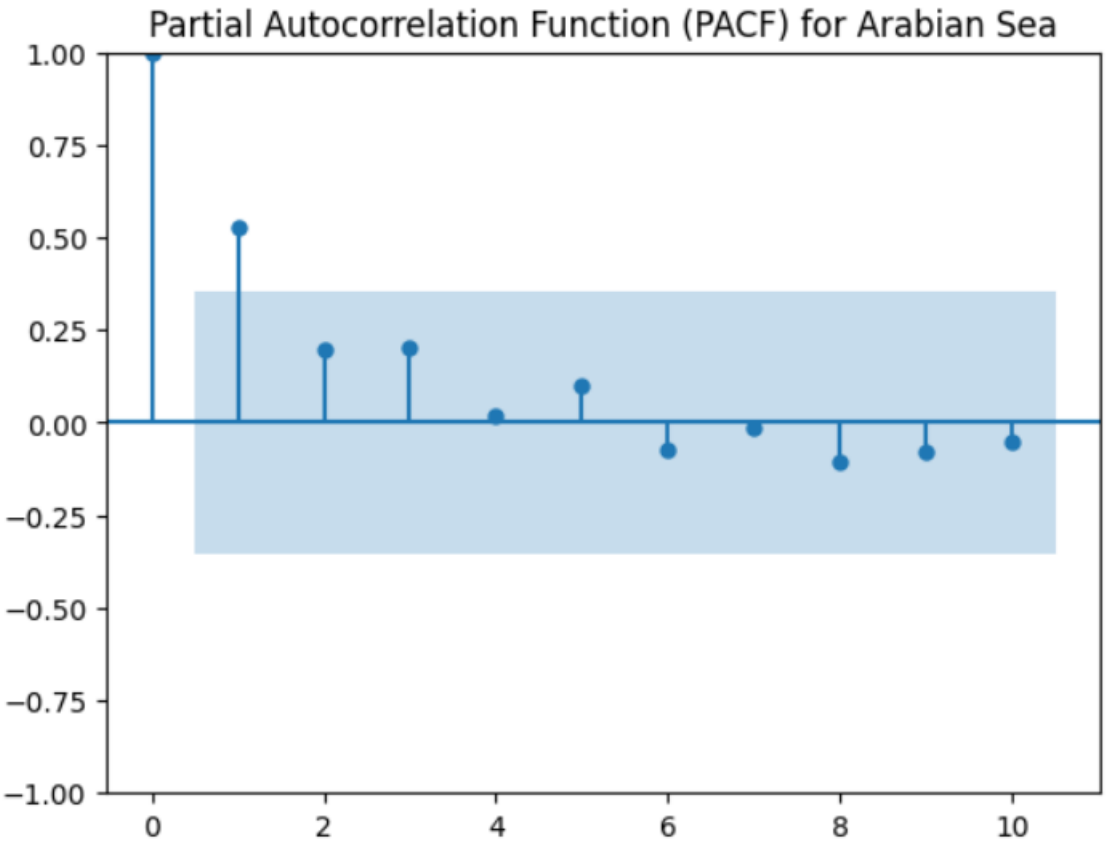

Partial Autocorrelation Function (PACF) for Arabian Sea
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10

Figure 6.23

Using AUTO ARIMA model to find (p,d,q):

Indian Ocean:

```
Best model:  ARIMA(2,1,0)(0,0,0)[0] intercept
Total fit time: 11.059 seconds
                               SARIMAX Results
==========================================================================================
Dep. Variable:                           y   No. Observations:                   31
Model:               SARIMAX(2, 1, 0)   Log Likelihood                -119.724
Date:                Tue, 10 Dec 2024   AIC                            247.449
Time:                        07:04:13   BIC                            253.053
Sample:                             0   HQIC                           249.242
                                  - 31
Covariance Type:                  opg
==========================================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------------------
intercept      7.4877      3.107      2.410      0.016       1.397      13.578
ar.L1         -0.7799      0.246     -3.174      0.002      -1.262      -0.298
ar.L2         -0.5269      0.204     -2.589      0.010      -0.926      -0.128
sigma2       166.0032     48.897      3.395      0.001      70.167     261.840
==========================================================================================
Ljung-Box (L1) (Q):                   0.04   Jarque-Bera (JB):                 0.29
Prob(Q):                              0.83   Prob(JB):                         0.87
Heteroskedasticity (H):               0.38   Skew:                             0.24
Prob(H) (two-sided):                  0.14   Kurtosis:                         3.02
==========================================================================================
```

Table 6.8

Using (p,d,q)=(2,1,0) values to build ARIMA

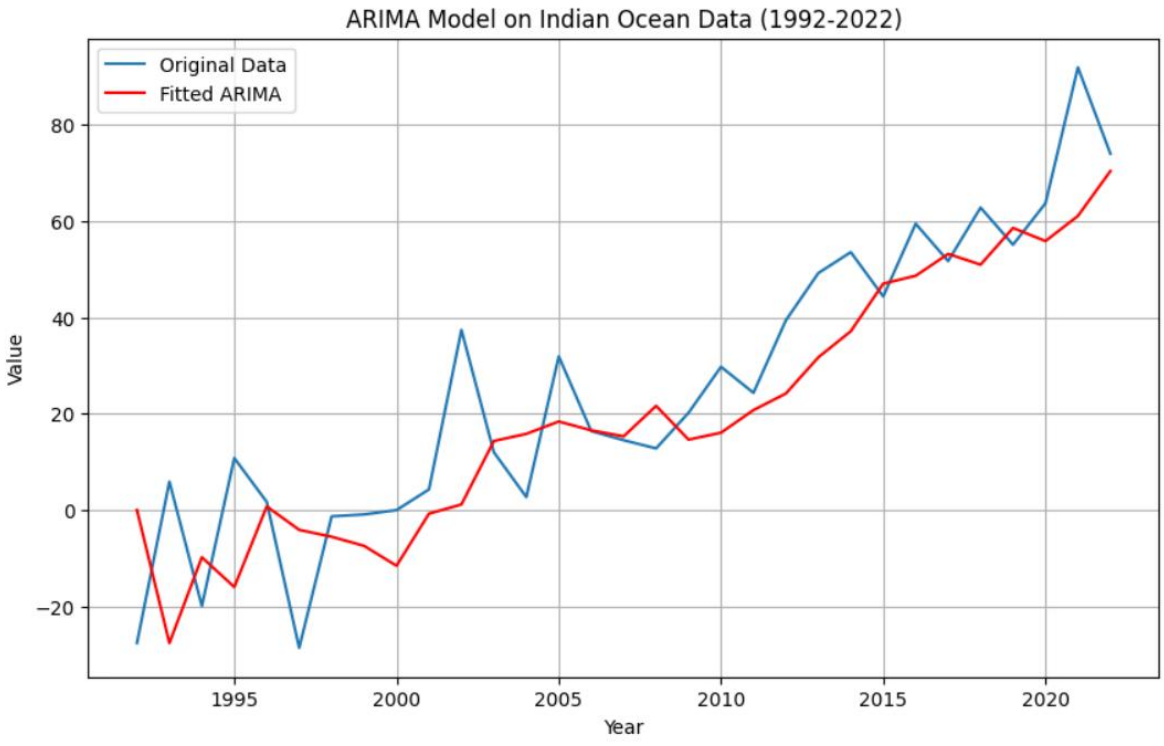
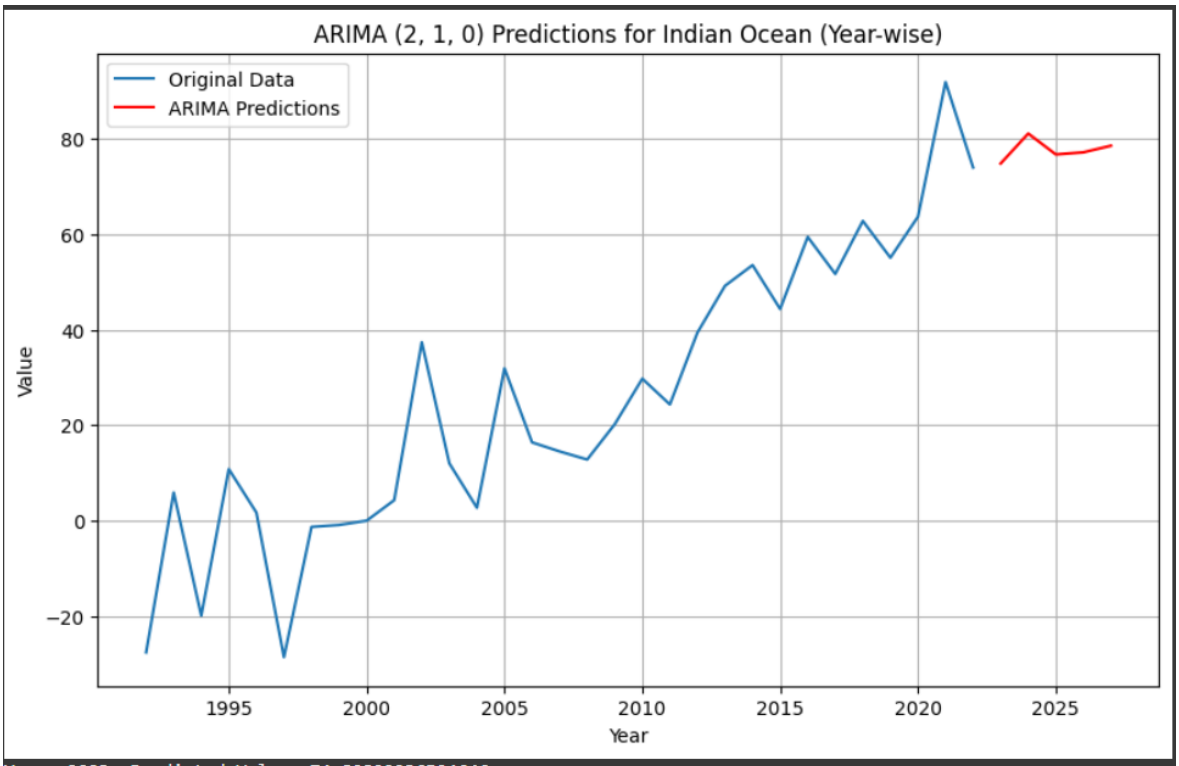


Figure 6.24




```
Year: 2023, Predicted Value: 74.80899936594646
Year: 2024, Predicted Value: 81.08379345353151
Year: 2025, Predicted Value: 76.70509012136324
Year: 2026, Predicted Value: 77.14779710312341
Year: 2027, Predicted Value: 78.52911964901244
```

Figure 6.25

Using AUTO ARIMA model to find (p,d,q):

Bay of Bengal:

```
Best model:  ARIMA(1,0,0)(0,1,0)[12] intercept
Total fit time: 13.378 seconds
                                SARIMAX Results
==========================================================================================
Dep. Variable:                                   y   No. Observations:                   31
Model:             SARIMAX(1, 0, 0)x(0, 1, 0, 12)   Log Likelihood                -99.040
Date:                            Tue, 10 Dec 2024   AIC                            204.081
Time:                                    07:04:58   BIC                            206.914
Sample:                                         0   HQIC                           204.560
                                              - 31
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     30.4637     13.975      2.180      0.029       3.073      57.854
ar.L1          0.3213      0.277      1.158      0.247      -0.223       0.865
sigma2      1962.3022    948.219      2.069      0.039     103.827    3820.777
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):                 1.33
Prob(Q):                              0.90   Prob(JB):                         0.51
Heteroskedasticity (H):               1.10   Skew:                             0.50
Prob(H) (two-sided):                  0.91   Kurtosis:                         2.17
===================================================================================
```

Table 6.9

Using (p,d,q)=(1,0,0) values to build ARIMA

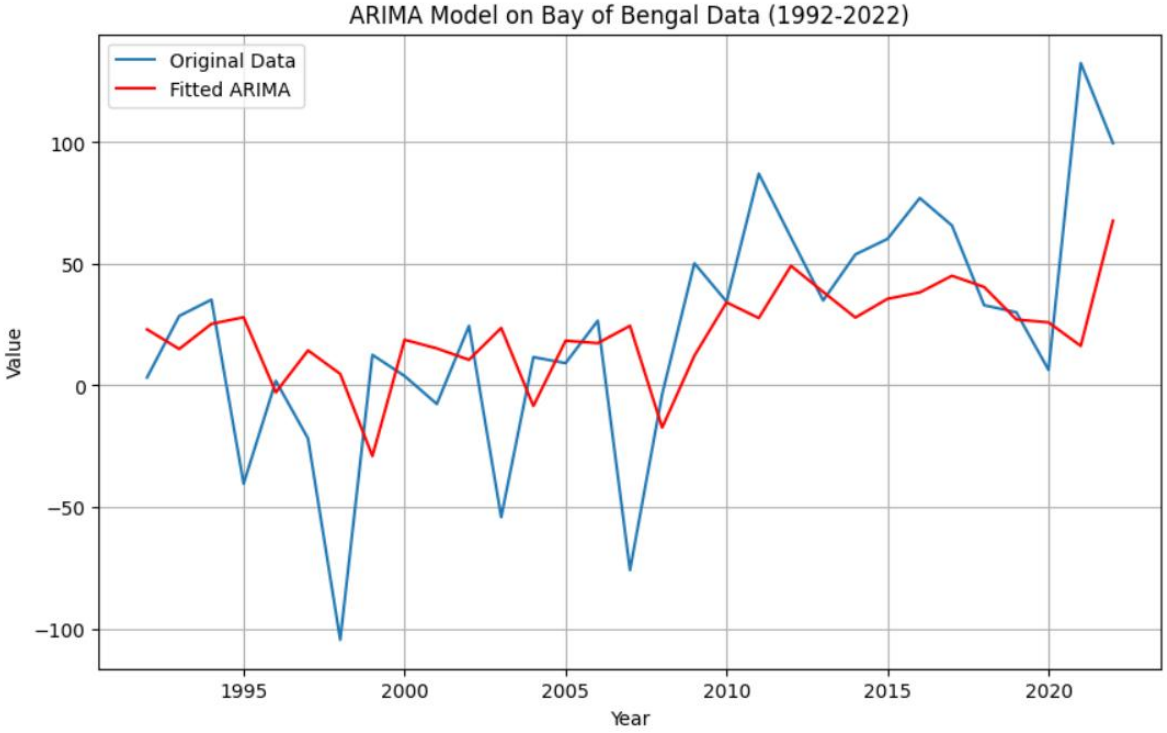
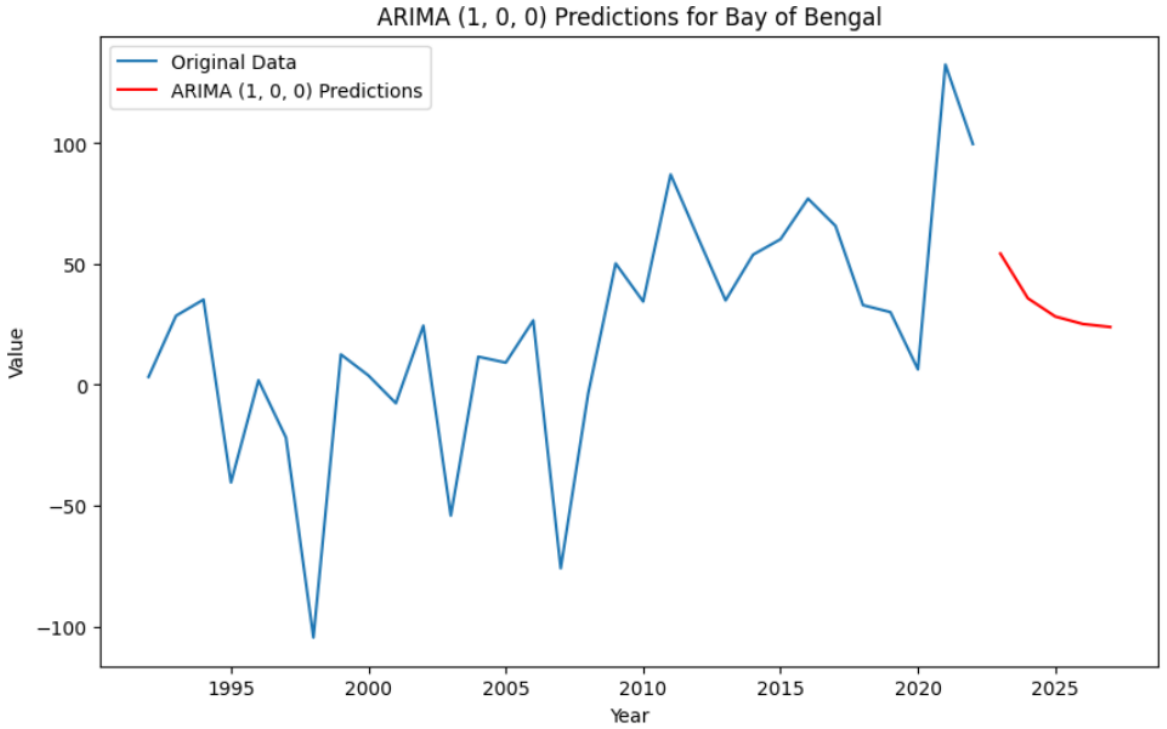


Figure 6.26




```
Predicted value for 2023: 54.3249113938531
Predicted value for 2024: 35.814727471378234
Predicted value for 2025: 28.2604062392197
Predicted value for 2026: 25.177359116746214
Predicted value for 2027: 23.919115090939002
```

Figure 6.27

Using AUTO ARIMA model to find (p,d,q):

Arabian Sea:

SARIMAX Results

Dep. Variable:	y	No. Observations:	31
Model:	SARIMAX(0, 1, 1)	Log Likelihood	-138.910
Date:	Tue, 10 Dec 2024	AIC	283.821
Time:	07:05:39	BIC	288.024
Sample:	0	HQIC	285.166
	- 31		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	2.9022	1.479	1.962	0.050	0.003	5.801
ma.L1	-0.7161	0.181	-3.951	0.000	-1.071	-0.361
sigma2	601.1656	165.203	3.639	0.000	277.374	924.958

Ljung-Box (L1) (Q):	0.05	Jarque-Bera (JB):	0.43
Prob(Q):	0.82	Prob(JB):	0.81
Heteroskedasticity (H):	0.85	Skew:	-0.28
Prob(H) (two-sided):	0.80	Kurtosis:	2.84

Table 6.10

Using (p,d,q)=(0,1,1) values to build ARIMA

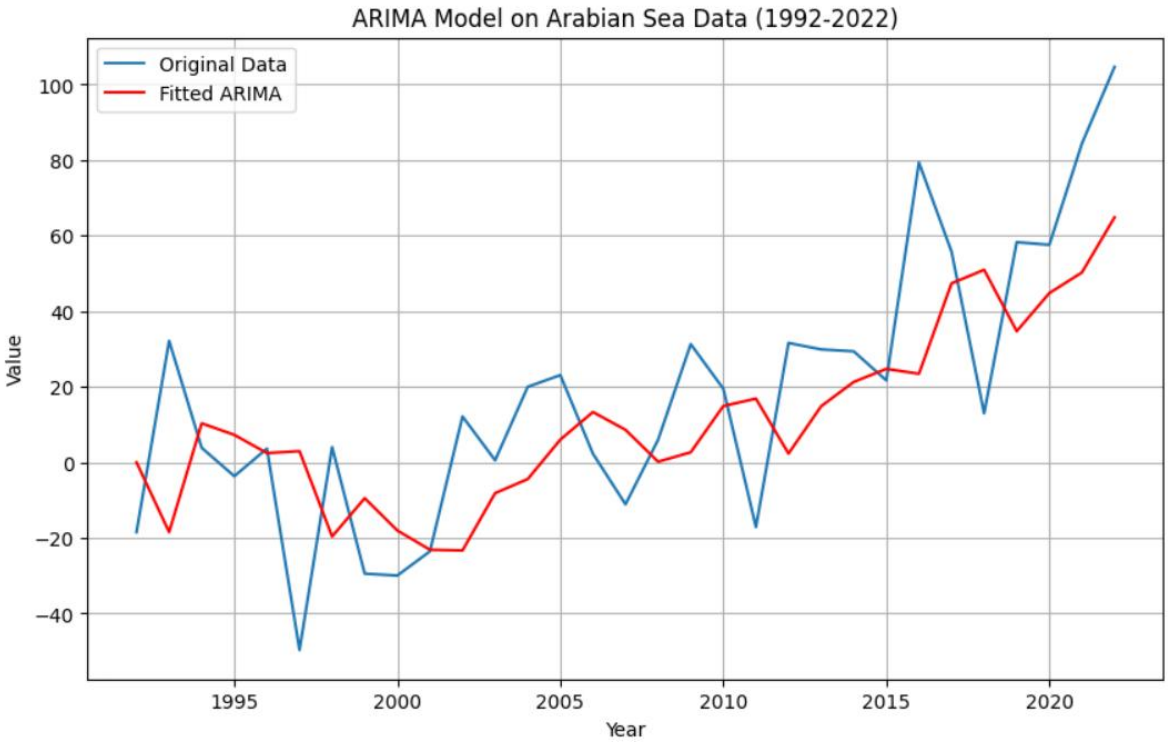
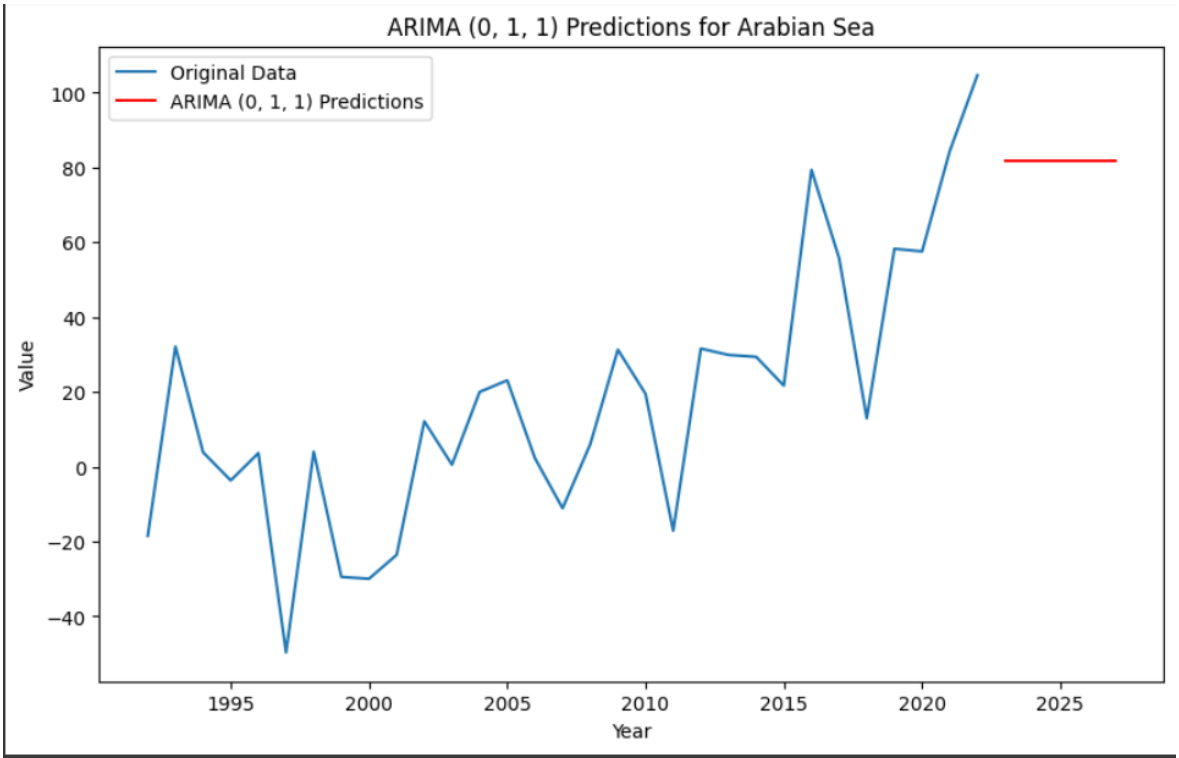


Figure 6.28




```
Predictions for Arabian Sea:
Predicted value for 2023: 81.87949659111074
Predicted value for 2024: 81.87949659111074
Predicted value for 2025: 81.87949659111074
Predicted value for 2026: 81.87949659111074
Predicted value for 2027: 81.87949659111074
```

Figure 6.29

Predicting using LSTM (Long Short-Term Memory) Model:

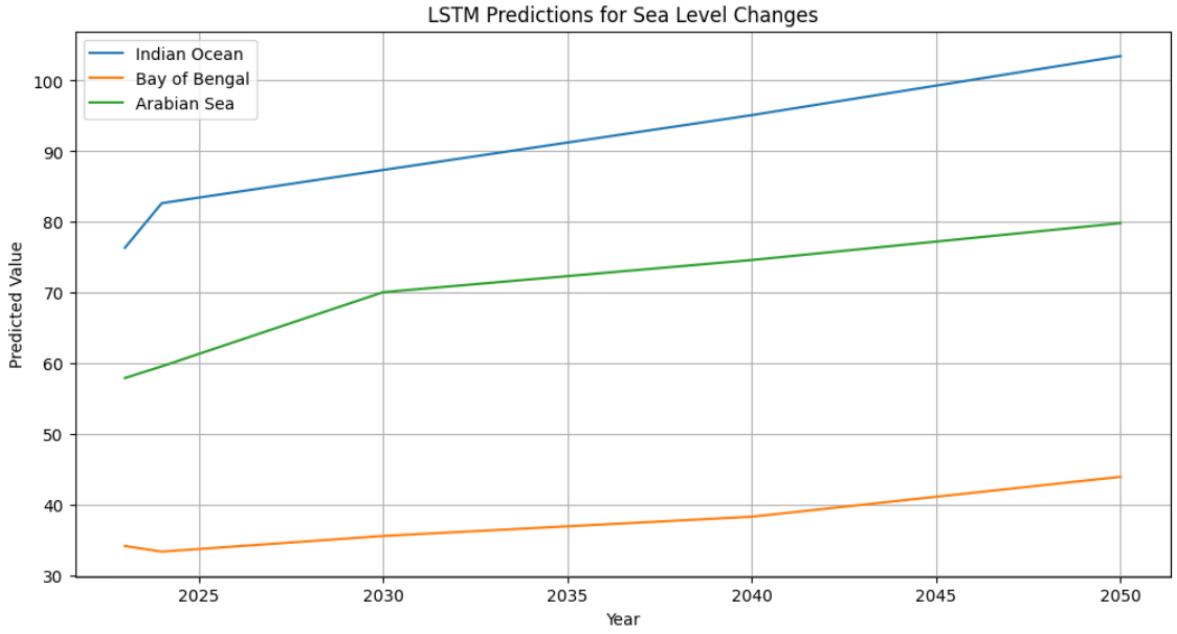


Figure 6.30

LSTM Model summary:

LSTM Model Summary for Indian Ocean:
Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 50)	10,400
dense (Dense)	(None, 1)	51

Total params: 31,355 (122.48 KB)
Trainable params: 10,451 (40.82 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 20,904 (81.66 KB)

Table 6.11

LSTM Model Summary for Bay of Bengal:
Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 50)	10,400
dense_1 (Dense)	(None, 1)	51

Total params: 31,355 (122.48 KB)
Trainable params: 10,451 (40.82 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 20,904 (81.66 KB)

Table 6.12

LSTM Model Summary for Arabian Sea:
Model: "sequential_2"

Layer (type)	Output Shape	Param #
lstm_2 (LSTM)	(None, 50)	10,400
dense_2 (Dense)	(None, 1)	51

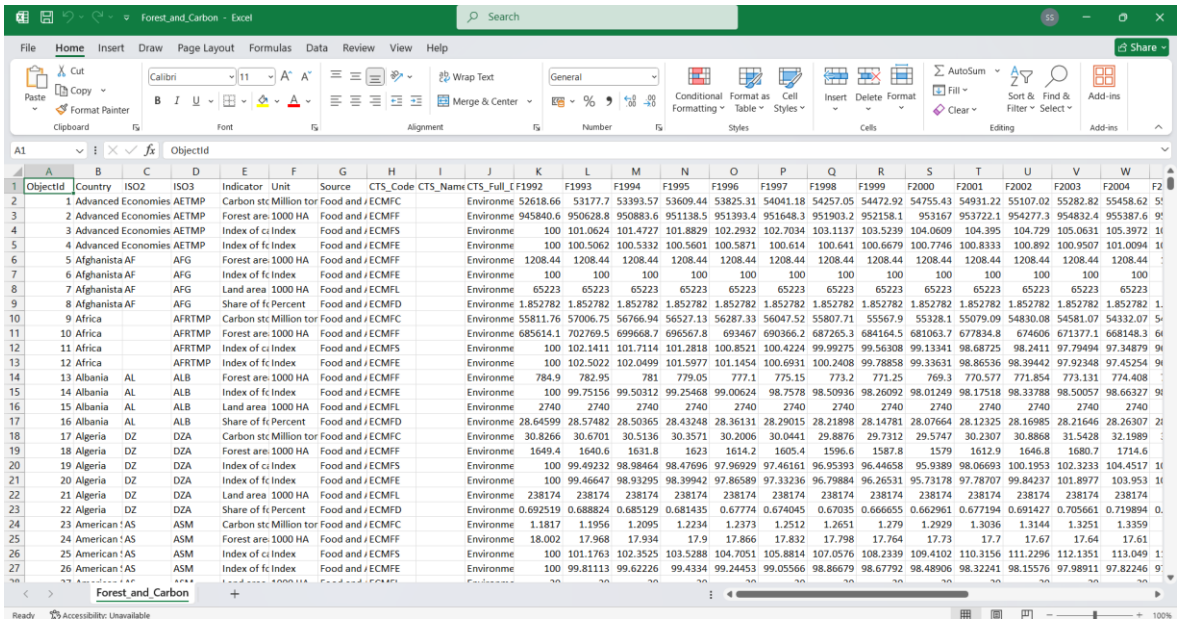
Total params: 31,355 (122.48 KB)
Trainable params: 10,451 (40.82 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 20,904 (81.66 KB)

Table 6.13

6.4 Forest Area Cover Dataset

6.4.1. ABOUT THE DATASET:

- **Source of the Dataset:** https://climatedata.imf.org/pages/access-data



Key Columns in the Dataset:

1. **ObjectId**: A unique identifier for each record.
2. **Country**: The name of the country or region for which the data is recorded (e.g., Advanced Economies, Afghanistan, Africa, Albania, Algeria).
3. **ISO2:** The two-letter ISO country code (e.g., AF for Afghanistan, DZ for Algeria).
4. **ISO3:** The three-letter ISO country code (e.g., AFG for Afghanistan, DZA for Algeria).
5. **Indicator:** The specific environmental indicator being measured (e.g., Carbon stocks in forests, Forest area, Share of forest area).
6. **Unit:** The unit of measurement for the data (e.g., Million tonnes, 1000 HA, Percent).
7. **Source:** The source of the data, which is the Food and Agriculture Organization of the United Nations (FAO) along with additional notes about data access and calculations.
8. **CTS_Code:** A code related to the environmental sector the data falls under (e.g., ECMFC, ECMFF).
9. **CTS_Name**: A descriptor of the environmental category (e.g., Environment, Climate Change, Mitigation, Forest and Carbon).

10. **CTS_Full_Descriptor:** A more detailed explanation of the environmental category (e.g., Carbon Stocks in Living Biomass (in Forests)).
11. **Yearly Data** (1992-2022): The data columns spanning multiple years (e.g., F1992, F1993, ..., F2022) contain the recorded values for each year corresponding to the respective indicator.

6.4.2. Exploratory Data Analysis

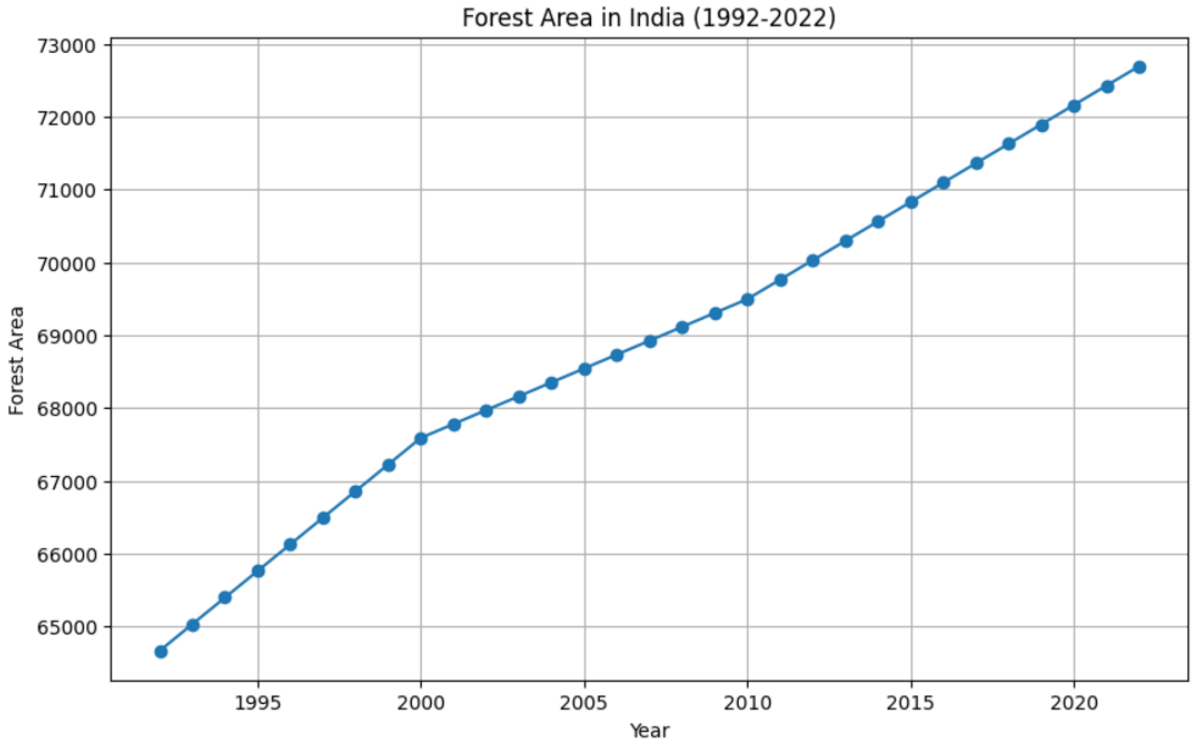


Figure 6.31

Interpretation: Forest area in India values seems to be increasing over the period of years plotted above.

Results and Discussion: Forest Area Cover

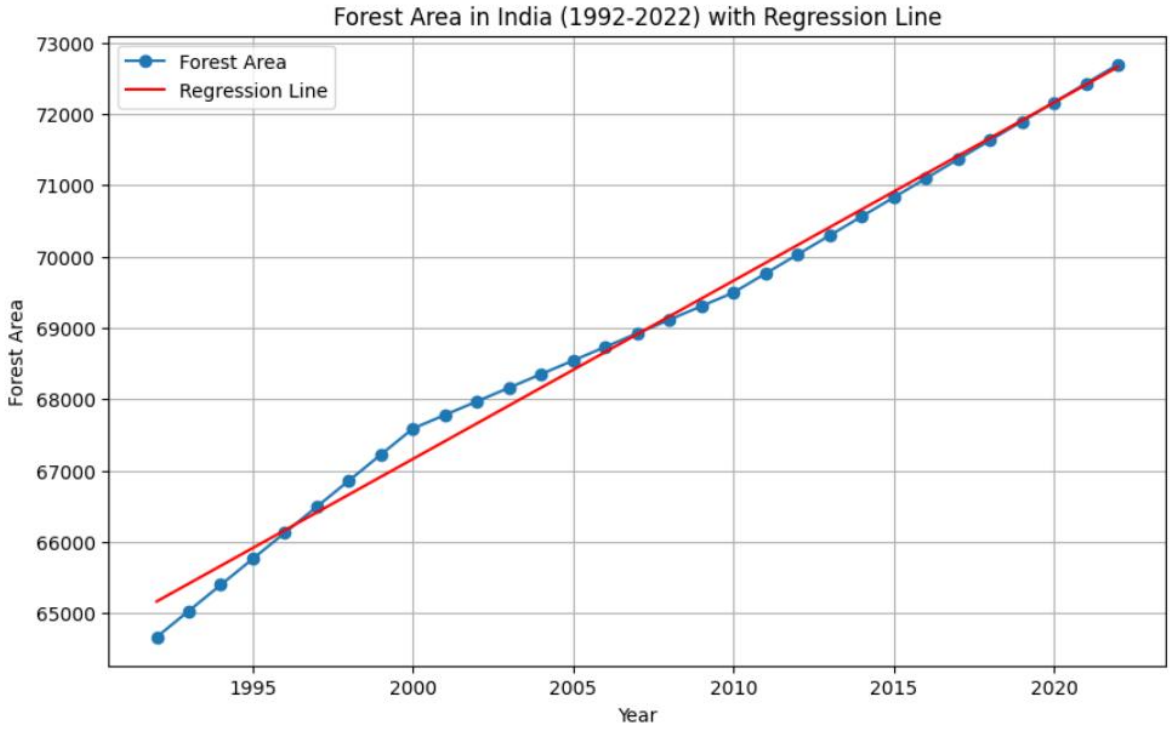


Figure 6.32

EVALUATION METRICS:

```
Slope: 249.69532258064515
Intercept: -432226.0317741935
R-value: 99.58763709339708%
P-value: 8.753224918248456e-32
Standard Error: 4.223889949234077
R-squared: 0.9917697461846157
```

PREDICTION USING LINEAR REGRESSION:

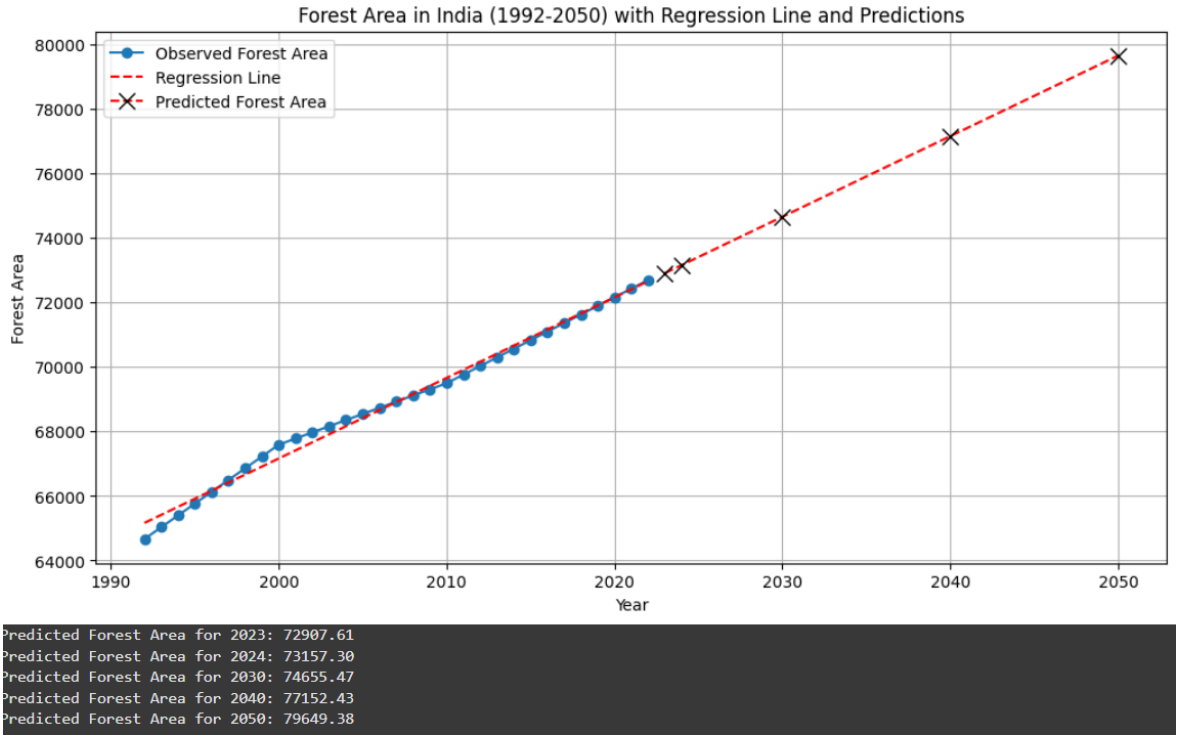


Figure 6.33

ACF and PACF Plots: Forest Area Cover:

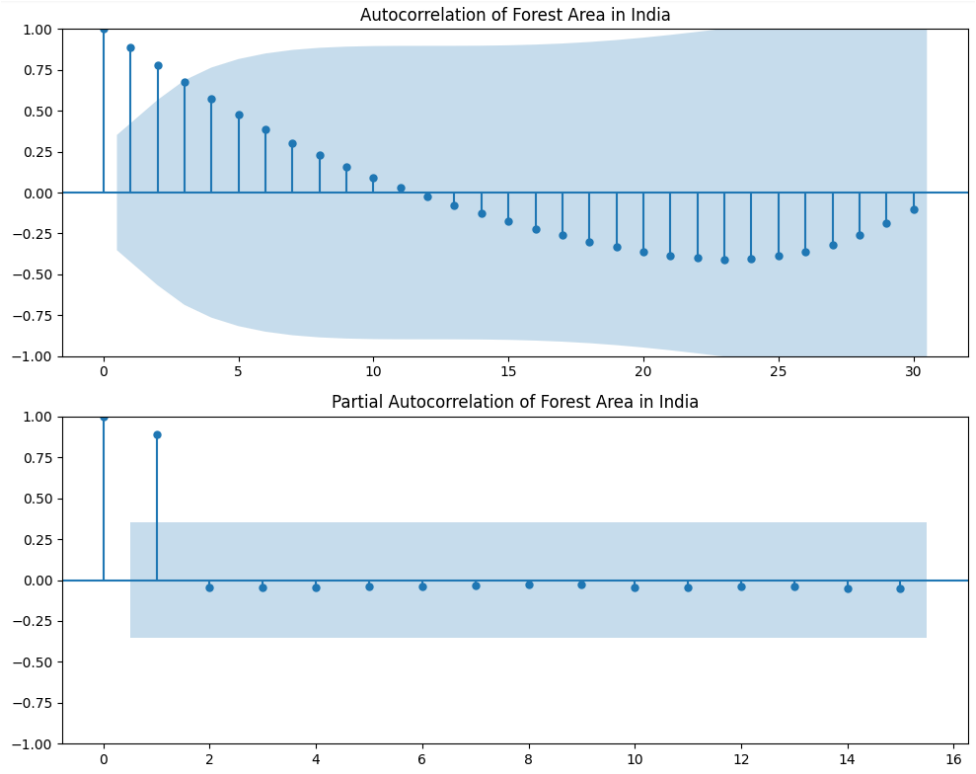


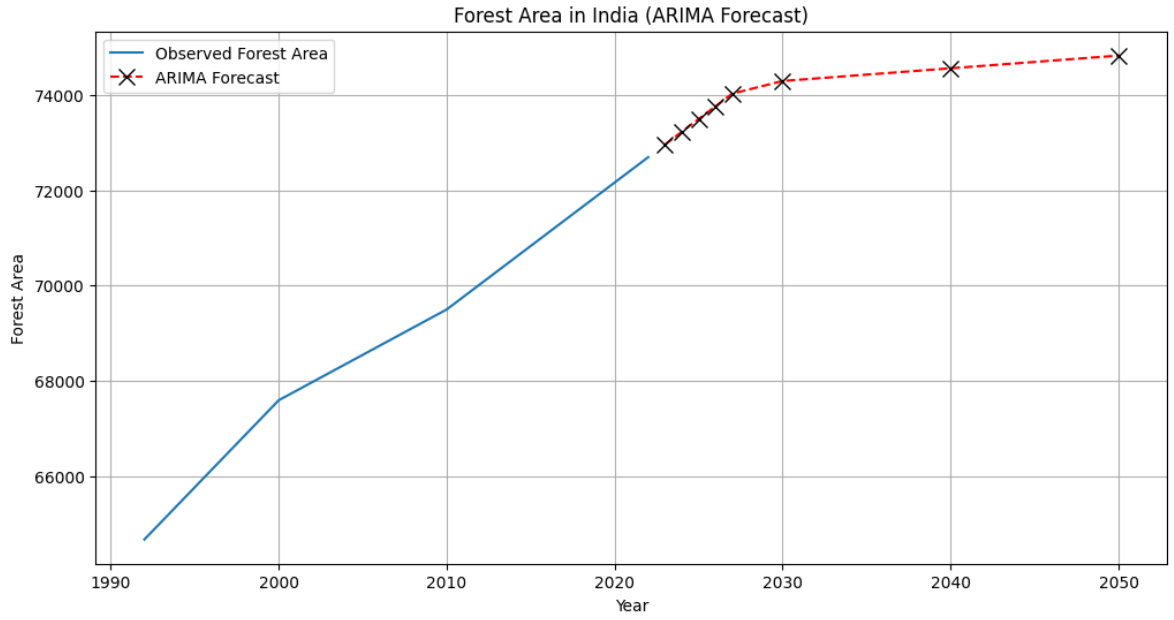
Figure 6.34

Using AUTO ARIMA model to find (p,d,q):

```
Best model:  ARIMA(0,2,0)(0,0,0)[0]
Total fit time: 0.315 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   31
Model:               SARIMAX(0, 2, 0)   Log Likelihood                -144.574
Date:                Tue, 10 Dec 2024   AIC                            291.147
Time:                        07:46:43   BIC                            292.514
Sample:                             0   HQIC                           291.575
                                  - 31
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2      1252.2601    103.327     12.119      0.000    1049.743    1454.777
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):               420.07
Prob(Q):                              0.96   Prob(JB):                         0.00
Heteroskedasticity (H):               0.00   Skew:                            -3.58
Prob(H) (two-sided):                  0.00   Kurtosis:                        20.22
===================================================================================
```

Table 6.14

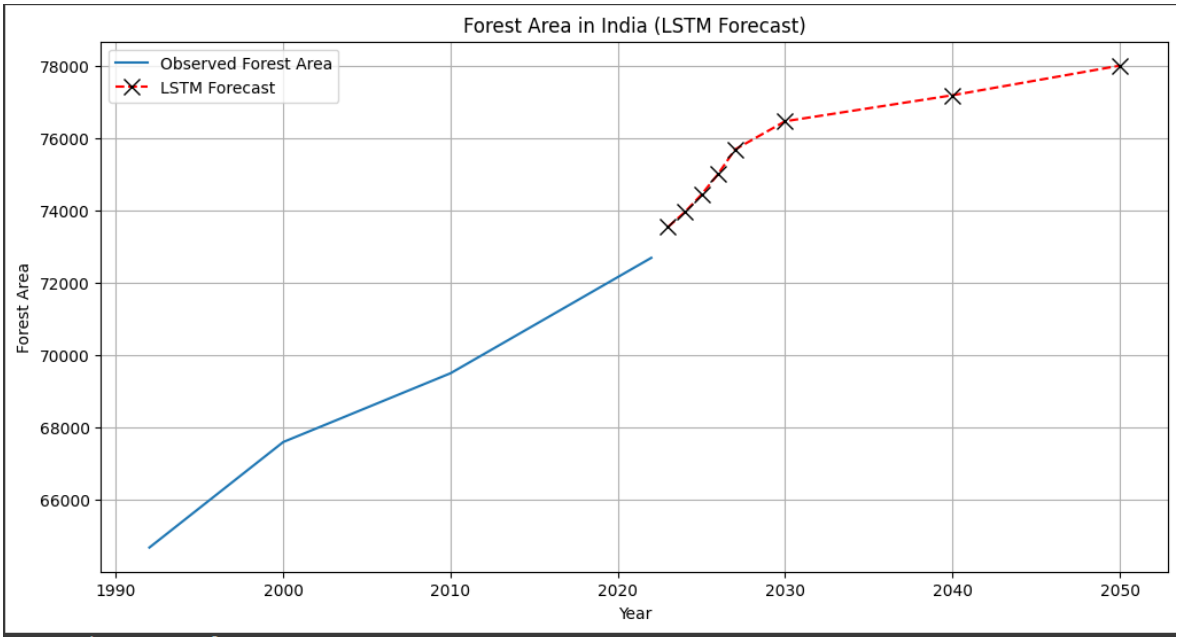
Using (p,d,q)=(0,2,0) values to build ARIMA




```
Forecasted Forest Area for 2023: 72959.20
Forecasted Forest Area for 2024: 73225.60
Forecasted Forest Area for 2025: 73492.00
Forecasted Forest Area for 2026: 73758.40
Forecasted Forest Area for 2027: 74024.80
Forecasted Forest Area for 2030: 74291.20
Forecasted Forest Area for 2040: 74557.60
Forecasted Forest Area for 2050: 74824.00
```

Figure 6.35

Predicting using LSTM (Long Short-Term Memory) Model :




```
Forecasted Forest Area for 2023: 73548.52
Forecasted Forest Area for 2024: 73956.98
Forecasted Forest Area for 2025: 74443.89
Forecasted Forest Area for 2026: 75018.81
Forecasted Forest Area for 2027: 75692.16
Forecasted Forest Area for 2030: 76475.04
Forecasted Forest Area for 2040: 77195.33
Forecasted Forest Area for 2050: 78015.30
```

Figure 6.36

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 5, 50)	10,400
lstm_1 (LSTM)	(None, 50)	20,200
dense (Dense)	(None, 1)	51

Total params: 91,955 (359.20 KB)
Trainable params: 30,651 (119.73 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 61,304 (239.47 KB)

Table 6.15

Performance Analysis: ARIMA vs LSTM

Strengths	**- Well-Suited for Short-Term Forecasts: Effective for short-term forecasting, especially if data is linear and has strong seasonal patterns.**	**- Handles Non-Linearities: Captures non-linear relationships and complex patterns in data, suitable for more complex datasets.**
	- Statistical Foundation: Has a solid statistical basis and is widely used in econometrics and other fields.	- Versatility: Applied to a wide range of time series problems, including those with irregular intervals and missing data.
Weaknesses	- Limited to Linear Relationships: Assumes linear relationships, which might not capture complex patterns.	- Complexity: More complex and requires more computational resources and time to train.
	- Struggles with Long-Term Forecasts: Can struggle with long-term forecasts due to reliance on past values and patterns that might not persist.	- Data Requirement: Generally, requires more data to train effectively compared to simpler models like ARIMA.
Practical Considerations	- **Short-Term Forecasts**: If data is linear and you need **short-term forecasts**, ARIMA can be a good choice.	**- Long-Term Forecasts**: If **data is non-linear** and you need long-term forecasts**, LSTM is likely the better option.**
	- Data Complexity: For simple and linear data, ARIMA might suffice.	- Data Complexity: For more complex data with multiple patterns and long-term dependencies, LSTM is preferred.

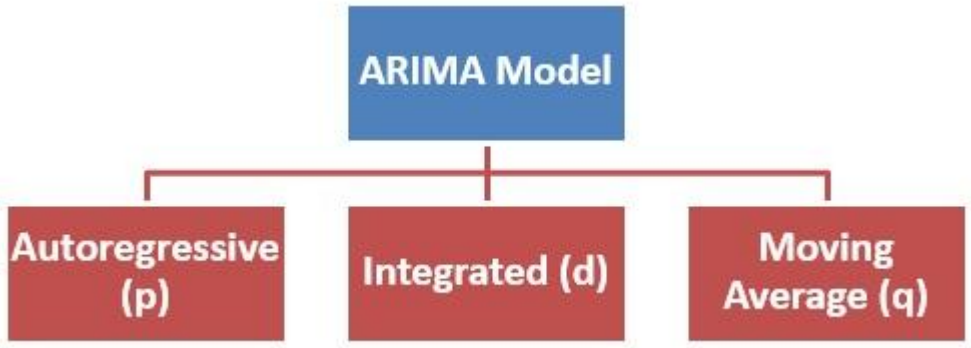
Table 6.16

6.5 MODELS USED IN STUDY

1. Linear Regression

- **Functionality**:
    - Linear Regression models the relationship between a dependent variable and one independent variable using a straight line.
    - It predicts the value of the dependent variable based on the value of the independent variable.
    - **Formula**: $Y=b0+b1*XY = b0 + b1 * X$
    - **Outputs**: Coefficient (slope), Intercept, R-squared value, p-value for significance testing.
- **Use Case**: To establish a basic understanding of the relationship between two variables, such as temperature change over time.

2. ARIMA (AutoRegressive Integrated Moving Average)

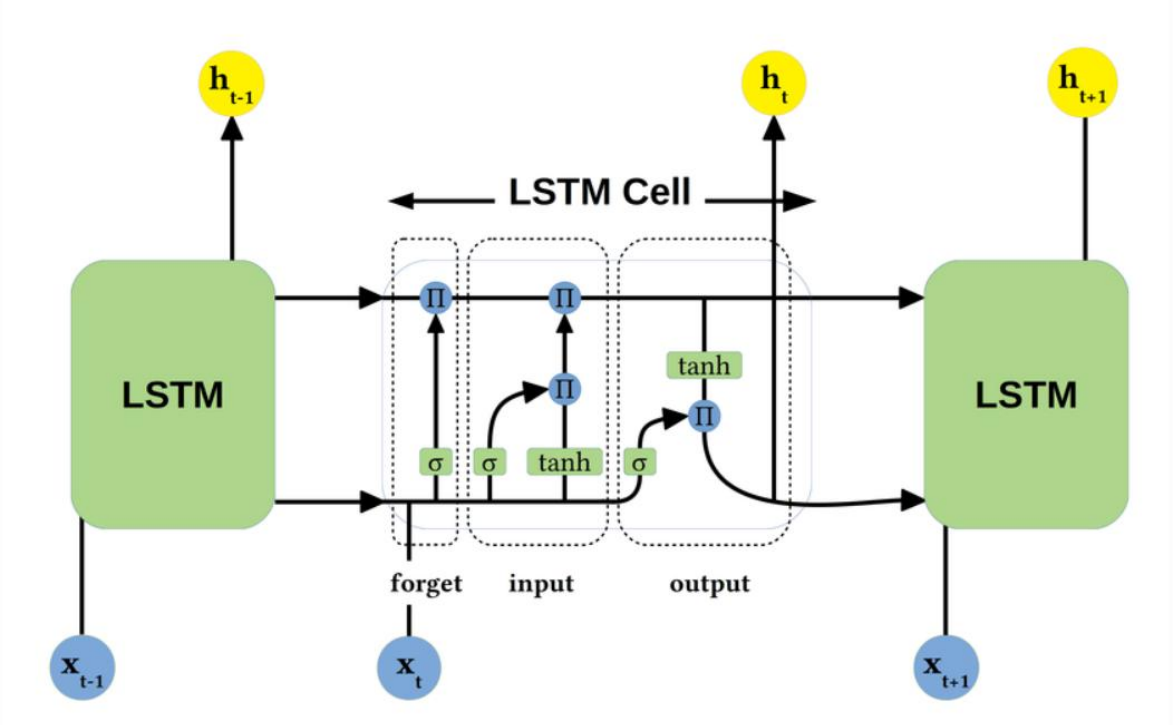


- **Functionality**:
    - ARIMA models time-series data by combining autoregression (AR), differencing (I), and moving average (MA).
    - **Components**:
        - **AutoRegressive (AR)**: Uses past values to predict future values.
        - **Integrated (I)**: Differencing the data to make it stationary.
        - **Moving Average (MA)**: Uses past forecast errors to improve the prediction.

- **Formula**: Yt=c+ϕ1∗Yt−1+ϕ2∗Yt−2+...+θ1∗ϵt−1+θ2∗ϵt−2+...+ϵtY_t = c + \phi_1 * Y_{t-1} + \phi_2 * Y_{t-2} + ... + \theta_1 * \epsilon_{t-1} + \theta_2 * \epsilon_{t-2} + ... + \epsilon_t
- **Outputs**: AR and MA coefficients, differencing order, model fit statistics (AIC, BIC), forecasts.

- **Use Case**: To capture and predict linear trends and seasonal patterns in time-series data, such as annual sea level changes.

4. LSTM (Long Short-Term Memory)



- **Functionality**:
  - LSTM is a type of recurrent neural network (RNN) designed to remember long-term dependencies in sequential data.
  - It uses memory cells to store information and gates to control the flow of information.
  - **Components**:
    - **Forget Gate**: Decides what information to discard from the cell state.
    - **Input Gate**: Decides which new information to add to the cell state.

        - **Output Gate**: Decides what to output based on the cell state.
    - **Formula**: Includes complex equations involving the gates and cell states, optimized during training.
    - **Outputs**: Predicted values, model loss (e.g., Mean Squared Error), training and validation accuracy.
- **Use Case**: To model complex, non-linear relationships and long-term dependencies in time-series data, such as predicting long-term temperature changes.

6.1.4. EVALUATION

Performance Metrics

- **Functionality**:
    - Metrics used to evaluate the accuracy and effectiveness of predictive models.
    - **Common Metrics**:
        - **Mean Squared Error (MSE)**: Average of the squared differences between predicted and actual values.
        - **Mean Absolute Error (MAE)**: Average of the absolute differences between predicted and actual values.
        - **Root Mean Squared Error (RMSE)**: Square root of the mean squared error, providing error in the same units as the original data.
        - **R-squared (R^2)**: Proportion of the variance in the dependent variable explained by the independent variables.
    - **Outputs**: Quantitative measures of model performance.
- **Use Case**: To compare the performance of different models and select the best one.

.

Relevance of COP 29 SUMMIT:

The *COP 29 summit, held in Baku from November 11 to 22, 2024*, brought several key findings and decisions to the forefront of global climate action. One of the most signif Table 7.2 Results of Machine learning models ilize **$300 billion annually by 2035 for climate finance to support developing countries. This decision aimed to address the urgent need for financial resources to combat climate change and its impacts.

However, ***India rejected this offer***, stating that the amount was grossly insufficient to meet the needs of developing nations. India's delegate, Chandni Raina, emphasized that the proposed $300 billion was an "optical illusion" and would not address the enormity of the climate challenges faced by the Global South. India argued that the financial commitment should be closer to **$1.3 trillion annually** to effectively tackle climate adaptation and transformation efforts.

India's rejection highlighted the ongoing equity concerns and the need for developed countries to take greater responsibility for their historical emissions. The decision underscored the importance of collaboration and trust among nations to achieve meaningful progress in addressing climate change.

Incorporating these findings into your project can provide valuable context for understanding the complexities of climate finance and the challenges faced by developing countries. It also emphasizes the need for accurate predictions and comprehensive climate policies to support equitable and effective climate action.

8.CONCLUSION

The ***primary goal of this project is to develop a sophisticated machine learning-based predictive model*** designed to estimate future values of key climate parameters: Atmospheric CO2 Concentrations, Sea Level Change, Land Cover Distribution, and Annual Surface Temperature Change. By leveraging historical trends and a range of relevant factors, this model aims to provide precise and reliable forecasts, which are critical for understanding and addressing the impacts of climate change. The ability to accurately predict these parameters will play a pivotal role in shaping effective and targeted climate policies and strategies.

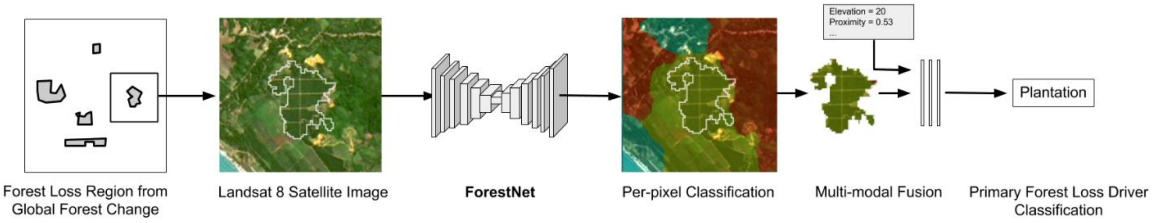
Additionally, the ***secondary goal of the project focuses on exploring and understanding the intricate relationships between these climate variables.*** By analyzing how changes in one parameter may influence others, this research seeks to provide deeper insights into the interconnected nature of climate systems. This understanding is essential for formulating more accurate and holistic climate projections. Such insights can significantly enhance the effectiveness of climate policies by enabling policymakers to account for the complex interactions within the climate system.

Furthermore, this project highlights the ***importance of international summits like COP 29***, held from November 11 to 22, 2024. These summits provide a critical platform for global leaders, scientists, and policymakers to discuss and negotiate climate action strategies. The insights gained from this project can inform these discussions, leading to more precise and targeted climate policies. By understanding the relationships between key climate variables and their future trajectories, stakeholders can make more informed decisions, ensuring that policies are based on robust scientific evidence and are capable of addressing the multifaceted challenges posed by climate change.

In conclusion, the development of a machine learning-based predictive model for climate parameters not only advances scientific understanding but also provides essential tools for policymakers. The project's dual focus on accurate prediction and understanding interrelationships between climate variables ensures that the resulting climate policies will be both comprehensive and effective. By contributing valuable insights to international climate negotiations, this research underscores the critical role of advanced technological approaches in tackling global environmental challenges

9. FUTURE ENHANCEMENT

- In the Forest Area Dataset instead of taking dataset from online sources I will use Satellite Imaging using **Forest Net [3]** which uses Deep Learning to predict actual deforestation and its impact.
- Developed by **STANFORD**: For accurate prediction of forest area.



10. APPENDICIES

CODES ARE ATTACHED DATABASE WISE:

- Annual Mean Global Surface Temperature Dataset
  PAGE 61-74
- Atmospheric Co2 concentration Dataset
  PAGE 75-87
- Change in Mean Sea Levels Dataset
  PAGE 88-106
- Forest Area Cover Dataset
  PAGE 107-124

Annual Mean Global Surface Temperature Dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd

# Assuming your dataset is in a CSV file named 'your_dataset.csv'
# Replace 'your_dataset.csv' with the actual filename and path
df = pd.read_csv('/content/drive/My Drive/Project sem1/Datasets/Annual_Mean_Global_Surface_Temperature.csv')

# Now you can work with the dataframe 'df'
print(df.head())
```

```
   ObjectId                       Country ISO2    ISO3  \
0         1  Afghanistan, Islamic Rep. of   AF     AFG
1         2                        Africa  NaN  AFRTMP
2         3                       Albania   AL     ALB
3         4                       Algeria   DZ     DZA
4         5                American Samoa   AS     ASM

                                          Indicator            Unit  \
0  Temperature change with respect to a baseline ...  Degree Celsius
1  Temperature change with respect to a baseline ...  Degree Celsius
2  Temperature change with respect to a baseline ...  Degree Celsius
3  Temperature change with respect to a baseline ...  Degree Celsius
4  Temperature change with respect to a baseline ...  Degree Celsius

                                            Source CTS Code  \
0  Food and Agriculture Organization of the Unite...     ECCS
1  Food and Agriculture Organization of the Unite...     ECCS
2  Food and Agriculture Organization of the Unite...     ECCS
3  Food and Agriculture Organization of the Unite...     ECCS
4  Food and Agriculture Organization of the Unite...     ECCS

                     CTS Name  \
0  Surface Temperature Change
1  Surface Temperature Change
2  Surface Temperature Change
3  Surface Temperature Change
4  Surface Temperature Change

                               CTS Full Descriptor  ...   2014   2015  \
0  Environment, Climate Change, Climate and Weath...  ...  0.521  1.204
1  Environment, Climate Change, Climate and Weath...  ...  1.013  1.190
2  Environment, Climate Change, Climate and Weath...  ...  1.285  1.667
3  Environment, Climate Change, Climate and Weath...  ...  1.713  1.153
4  Environment, Climate Change, Climate and Weath...  ...  0.840  0.679

    2016   2017   2018   2019   2020   2021   2022   2023
0  1.612  1.642  1.624  0.991  0.587  1.475  2.154  1.956
1  1.392  1.180  1.178  1.297  1.200  1.396  0.996  1.485
2  1.558  1.196  2.103  1.751  1.567  1.589  1.585  2.122
3  1.787  1.533  1.231  1.107  1.905  2.360  1.776  2.274
4  1.209  1.105  0.859  1.209  1.101  0.938  0.925  0.951

[5 rows x 73 columns]
```

```
df = df.drop('ISO2', axis=1)

print(df.head())
```

```
   ObjectId                       Country    ISO3  \
0         1  Afghanistan, Islamic Rep. of     AFG
1         2                        Africa  AFRTMP
2         3                       Albania     ALB
3         4                       Algeria     DZA
4         5                American Samoa     ASM

                                          Indicator            Unit  \
0  Temperature change with respect to a baseline ...  Degree Celsius
1  Temperature change with respect to a baseline ...  Degree Celsius
2  Temperature change with respect to a baseline ...  Degree Celsius
3  Temperature change with respect to a baseline ...  Degree Celsius
4  Temperature change with respect to a baseline ...  Degree Celsius

                                            Source CTS Code  \
0  Food and Agriculture Organization of the Unite...     ECCS
1  Food and Agriculture Organization of the Unite...     ECCS
2  Food and Agriculture Organization of the Unite...     ECCS
3  Food and Agriculture Organization of the Unite...     ECCS
4  Food and Agriculture Organization of the Unite...     ECCS

                     CTS Name  \
```

```
0  Surface Temperature Change
1  Surface Temperature Change
2  Surface Temperature Change
3  Surface Temperature Change
4  Surface Temperature Change

                               CTS Full Descriptor   1961  ...   2014  \
0  Environment, Climate Change, Climate and Weath... -0.126  ...  0.521
1  Environment, Climate Change, Climate and Weath... -0.017  ...  1.013
2  Environment, Climate Change, Climate and Weath...  0.635  ...  1.285
3  Environment, Climate Change, Climate and Weath...  0.155  ...  1.713
4  Environment, Climate Change, Climate and Weath...  0.121  ...  0.840

    2015   2016   2017   2018   2019   2020   2021   2022   2023
0  1.204  1.612  1.642  1.624  0.991  0.587  1.475  2.154  1.956
1  1.190  1.392  1.180  1.178  1.297  1.200  1.396  0.996  1.485
2  1.667  1.558  1.196  2.103  1.751  1.567  1.589  1.585  2.122
3  1.153  1.787  1.533  1.231  1.107  1.905  2.360  1.776  2.274
4  0.679  1.209  1.105  0.859  1.209  1.101  0.938  0.925  0.951

[5 rows x 72 columns]
```

```
df.dtypes
```

	0
ObjectId	int64
Country	object
ISO3	object
Indicator	object
Unit	object
...	...
2019	float64
2020	float64
2021	float64
2022	float64
2023	float64

72 rows × 1 columns

dtype: object

```
# Find all unique countries in the DataFrame
unique_countries = df['Country'].unique()

# Print the unique countries
unique_countries
```

```
array(['Afghanistan, Islamic Rep. of', 'Africa', 'Albania', 'Algeria',
       'American Samoa', 'Americas', 'Andorra, Principality of', 'Angola',
       'Anguilla', 'Antarctica', 'Antigua and Barbuda', 'Argentina',
       'Armenia, Rep. of', 'Aruba, Kingdom of the Netherlands', 'Asia',
       'Australia', 'Austria', 'Azerbaijan, Rep. of', 'Bahamas, The',
       'Bahrain, Kingdom of', 'Bangladesh', 'Barbados',
       'Belarus, Rep. of', 'Belgium', 'Belize', 'Benin', 'Bhutan',
       'Bolivia', 'Bosnia and Herzegovina', 'Botswana', 'Brazil',
       'British Virgin Islands', 'Brunei Darussalam', 'Bulgaria',
       'Burkina Faso', 'Burundi', 'Cabo Verde', 'Cambodia', 'Cameroon',
       'Canada', 'Cayman Islands', 'Central African Rep.', 'Chad',
       'Chile', 'China, P.R.: Hong Kong', 'China, P.R.: Macao',
       'China, P.R.: Mainland', 'Colombia', 'Comoros, Union of the',
       'Congo, Dem. Rep. of the', 'Congo, Rep. of', 'Cook Islands',
       'Costa Rica', "Côte d'Ivoire", 'Croatia, Rep. of', 'Cuba',
       'Cyprus', 'Czech Rep.', 'Denmark', 'Djibouti', 'Dominica',
       'Dominican Rep.', 'Ecuador', 'Egypt, Arab Rep. of', 'El Salvador',
       'Equatorial Guinea, Rep. of', 'Eritrea, The State of',
       'Estonia, Rep. of', 'Eswatini, Kingdom of',
       'Ethiopia, The Federal Dem. Rep. of', 'Europe',
       'Falkland Islands (Malvinas)', 'Faroe Islands', 'Fiji, Rep. of',
       'Finland', 'France', 'French Polynesia', 'Gabon', 'Gambia, The',
       'Georgia', 'Germany', 'Ghana', 'Gibraltar', 'Greece', 'Greenland',
       'Grenada', 'Guadeloupe', 'Guatemala', 'Guiana, French', 'Guinea',
       'Guinea-Bissau', 'Guyana', 'Haiti', 'Holy See', 'Honduras',
       'Hungary', 'Iceland', 'India', 'Indonesia',
       'Iran, Islamic Rep. of', 'Iraq', 'Ireland', 'Isle of Man',
       'Israel', 'Italy', 'Jamaica', 'Japan', 'Jordan',
       'Kazakhstan, Rep. of', 'Kenya', 'Kiribati',
       "Korea, Dem. People's Rep. of", 'Korea, Rep. of', 'Kuwait',
       'Kyrgyz Rep.', "Lao People's Dem. Rep.", 'Latvia', 'Lebanon',
```

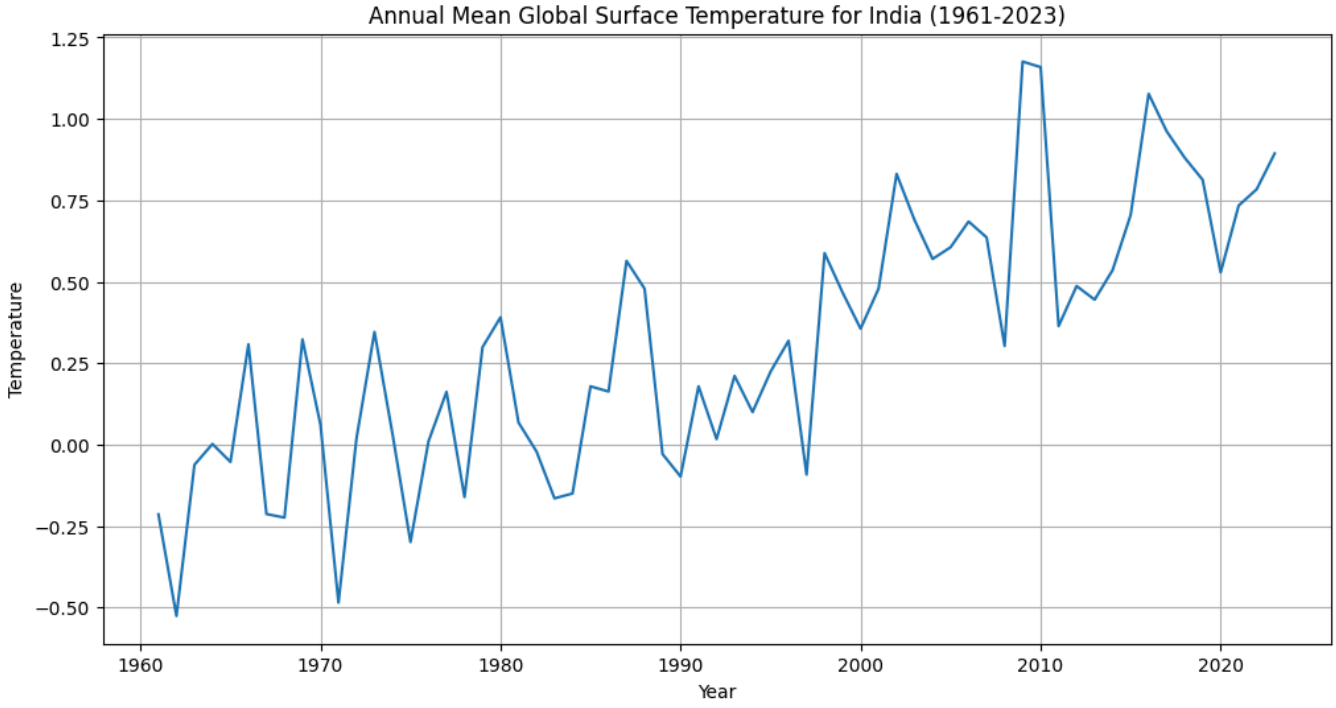
```
'Lesotho, Kingdom of', 'Liberia', 'Libya', 'Liechtenstein',
'Lithuania', 'Luxembourg', 'Madagascar, Rep. of', 'Malawi',
'Malaysia', 'Maldives', 'Mali', 'Malta',
'Marshall Islands, Rep. of the', 'Martinique',
'Mauritania, Islamic Rep. of', 'Mauritius', 'Mayotte', 'Mexico',
'Micronesia, Federated States of', 'Moldova, Rep. of', 'Monaco',
'Mongolia', 'Montenegro', 'Montserrat', 'Morocco',
'Mozambique, Rep. of', 'Myanmar', 'Namibia', 'Nauru, Rep. of',
'Nepal', 'Netherlands, The', 'New Caledonia', 'New Zealand',
'Nicaragua', 'Niger', 'Nigeria', 'Niue', 'Norfolk Island',
'North Macedonia, Republic of ', 'Norway', 'Oceania', 'Oman',
'Pakistan', 'Palau, Rep. of', 'Panama', 'Papua New Guinea',
'Paraguay', 'Peru', 'Philippines', 'Pitcairn Islands',
'Poland, Rep. of', 'Portugal', 'Puerto Rico', 'Qatar', 'Réunion',
'Romania', 'Russian Federation', 'Rwanda', 'Saint Helena',
'Saint Pierre and Miquelon', 'Samoa', 'San Marino, Rep. of',
'São Tomé and Príncipe, Dem. Rep. of', 'Saudi Arabia', 'Senegal',
'Serbia, Rep. of', 'Seychelles', 'Sierra Leone', 'Singapore',
'Slovak Rep.', 'Slovenia, Rep. of', 'Solomon Islands', 'Somalia',
'South Africa', 'South Sudan, Rep. of', 'Spain', 'Sri Lanka',
'St. Kitts and Nevis', 'St. Lucia',
'St. Vincent and the Grenadines', 'Sudan', 'Suriname',
'Svalbard Jan Mayen Islands', 'Sweden', 'Switzerland',
'Syrian Arab Rep.', 'Taiwan Province of China',
'Tajikistan, Rep. of', 'Tanzania, United Rep. of', 'Thailand',
'Timor-Leste, Dem. Rep. of', 'Togo', 'Tokelau', 'Tonga',
'Trinidad and Tobago', 'Tunisia', 'Türkiye, Rep. of'
```

```
# Filter the DataFrame for rows where 'Country' is 'India'
india_data = df[df['Country'] == 'India']

# Find and print the null/missing values in the filtered DataFrame
print(india_data.isnull().sum())
print(india_data.isnull().sum().sum())
```

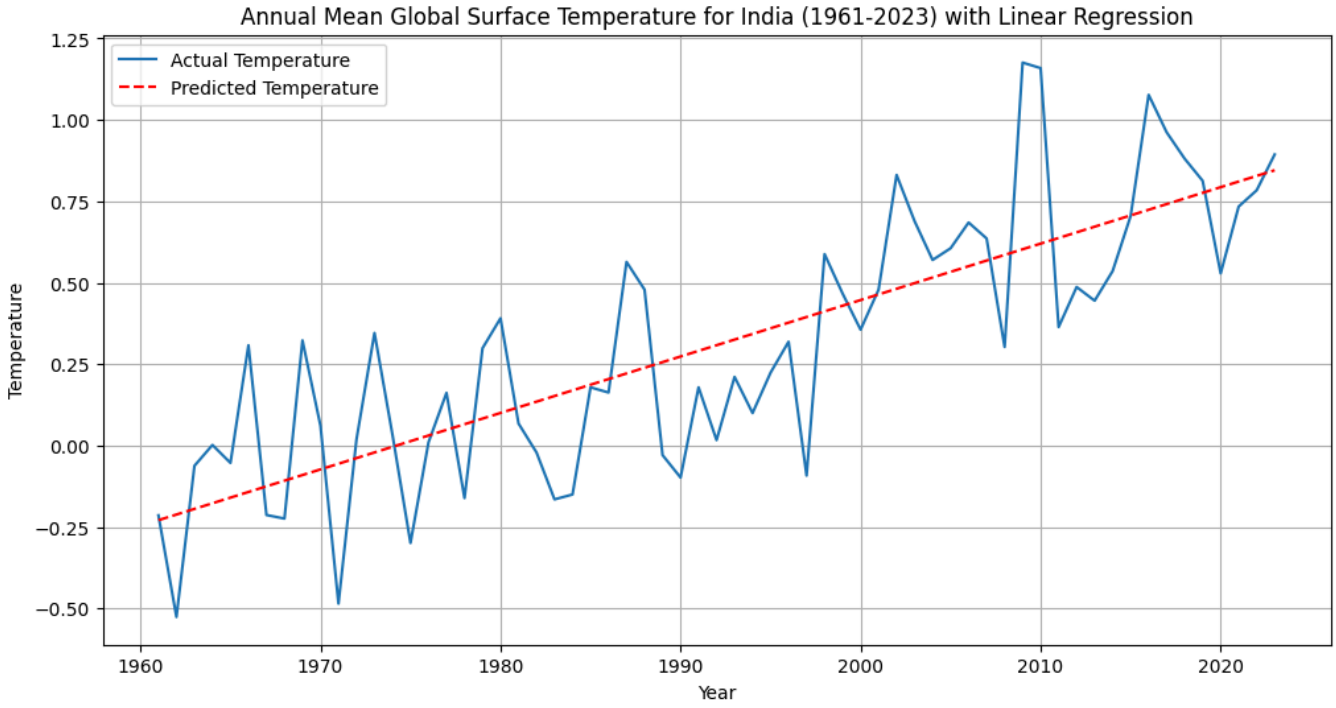
```
ObjectId     0
Country      0
ISO3         0
Indicator    0
Unit         0
            ..
2019         0
2020         0
2021         0
2022         0
2023         0
Length: 72, dtype: int64
0
```

```
import matplotlib.pyplot as plt

# Filter data for India
india_data = df[df['Country'] == 'India']
years = list(range(1961, 2024))
year_columns = [str(year) for year in years]

# Select temperatures for India for the specified years
temperatures = india_data.loc[india_data['Country'] == 'India', year_columns].values.flatten()
print(temperatures)

# Create the plot
plt.figure(figsize=(12, 6))
plt.plot(years, temperatures)
plt.xlabel('Year')
plt.ylabel('Temperature')
plt.title('Annual Mean Global Surface Temperature for India (1961-2023)')
plt.grid(True)
plt.show()
```

```
[-0.214 -0.526 -0.062  0.002 -0.053  0.308 -0.213 -0.224  0.323  0.061
 -0.485  0.019  0.346  0.033 -0.299  0.01   0.162 -0.161  0.299  0.391
  0.068 -0.021 -0.165 -0.15   0.179  0.163  0.564  0.479 -0.029 -0.098
  0.179  0.017  0.211  0.1    0.224  0.319 -0.092  0.588  0.467  0.356
  0.479  0.831  0.689  0.57   0.606  0.685  0.636  0.303  1.176  1.159
  0.364  0.487  0.445  0.536  0.705  1.077  0.962  0.881  0.813  0.529
  0.734  0.784  0.894]
```

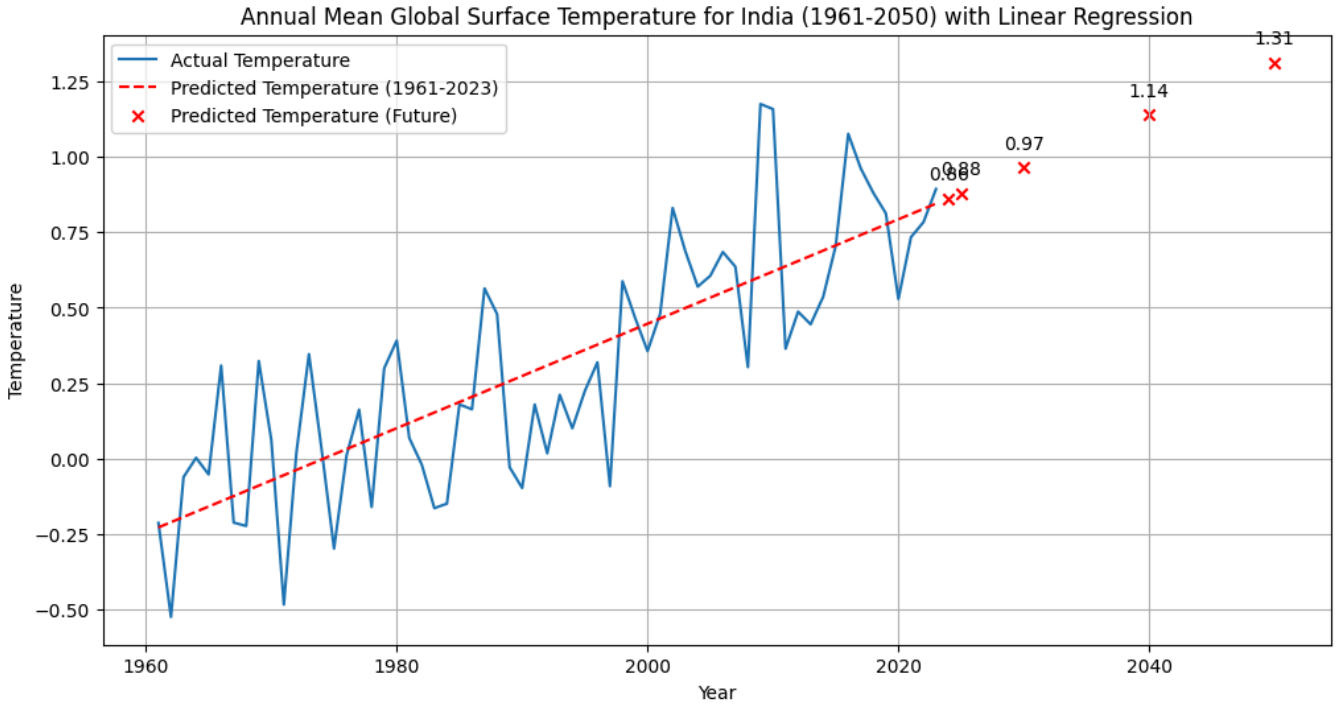



```
from sklearn.linear_model import LinearRegression

# Prepare the data for regression
X = [[year] for year in years]
y = temperatures

# Create and train the linear regression model
model = LinearRegression()
model.fit(X, y)

# Make predictions
y_pred = model.predict(X)

# Plot the actual and predicted temperatures
plt.figure(figsize=(12, 6))
plt.plot(years, temperatures, label='Actual Temperature')
plt.plot(years, y_pred, label='Predicted Temperature', linestyle='--',color='red')
plt.xlabel('Year')
plt.ylabel('Temperature')
plt.title('Annual Mean Global Surface Temperature for India (1961-2023) with Linear Regression')
plt.legend()
plt.grid(True)
plt.show()

# Print the model's coefficients
print(f"Intercept: {model.intercept_}")
print(f"Slope: {model.coef_[0]}")
```




```
Intercept: -34.21007232462878
Slope: 0.017328485023041476
```

```
# Predict temperatures for future years
future_years = [2024, 2025, 2030, 2040, 2050]
future_X = [[year] for year in future_years]
future_y_pred = model.predict(future_X)

# Plot the actual and predicted temperatures along with future predictions
plt.figure(figsize=(12, 6))
plt.plot(years, temperatures, label='Actual Temperature')
plt.plot(years, y_pred, label='Predicted Temperature (1961-2023)', linestyle='--', color='red')
plt.scatter(future_years, future_y_pred, color='red', label='Predicted Temperature (Future)', marker='x')
plt.xlabel('Year')
plt.ylabel('Temperature')
plt.title('Annual Mean Global Surface Temperature for India (1961-2050) with Linear Regression')
plt.legend()
plt.grid(True)

# Annotate the predicted future temperatures on the plot
for i, txt in enumerate(future_y_pred):
    plt.annotate(f'{txt:.2f}', (future_years[i], future_y_pred[i]), textcoords="offset points", xytext=(0,10), ha='center')

plt.show()

# Print the predicted future temperatures
for year, temp in zip(future_years, future_y_pred):
    print(f"Predicted temperature for {year}: {temp:.2f}")
```

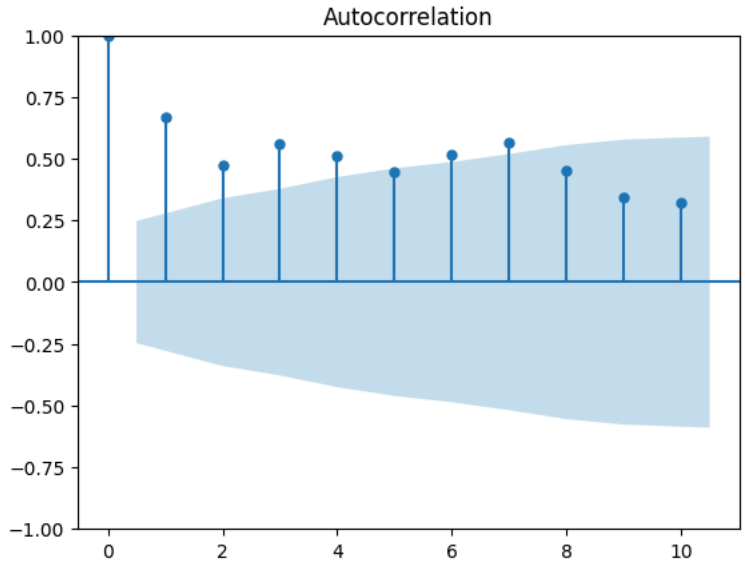


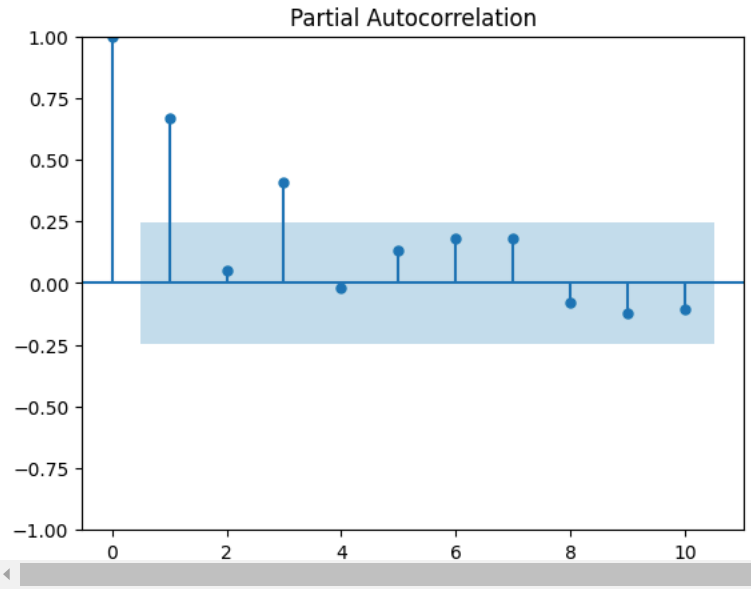

```
Predicted temperature for 2024: 0.86
Predicted temperature for 2025: 0.88
Predicted temperature for 2030: 0.97
Predicted temperature for 2040: 1.14
Predicted temperature for 2050: 1.31
```

```
import numpy as np
from sklearn.metrics import mean_squared_error,mean_absolute_error
from scipy import stats

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y, y_pred)
print(f"Mean Squared Error (MSE): {mse}")

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y, y_pred)
print(f"Mean Absolute Error (MAE): {mae}")
```

```
Mean Squared Error (MSE): 0.05746845380453541
Root Mean Squared Error (RMSE): 0.23972578877654238
Mean Absolute Error (MAE): 0.19309536559546114
```

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Plot ACF
plot_acf(temperatures, lags=10)  # Adjust lags as needed
plt.show()

# Plot PACF
plot_pacf(temperatures, lags=10)  # Adjust lags as needed
plt.show()
```






```
from statsmodels.tsa.stattools import adfuller

# Perform the ADF test
result = adfuller(temperatures, autolag='AIC')

print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

# Interpret the results
if result[1] <= 0.05:
    print("Strong evidence against the null hypothesis")
    print("Reject the null hypothesis")
    print("Data is stationary")
else:
    print("Weak evidence against the null hypothesis")
    print("Fail to reject the null hypothesis")
    print("Data is non-stationary")
```

```
ADF Statistic: -0.333389
p-value: 0.920641
Critical Values:
        1%: -3.553
        5%: -2.915
        10%: -2.595
Weak evidence against the null hypothesis
Fail to reject the null hypothesis
Data is non-stationary
```

```
best_d = None
best_p_value = 1  # Initialize with a high p-value
```

```
for d in range(4):  # Try differencing up to 3 times (0 to 3)
    differenced_data = temperatures
    for _ in range(d):
      differenced_data = np.diff(differenced_data)
    if len(differenced_data) < 2:
        break  # Stop if differencing results in too few data points

    result = adfuller(differenced_data, autolag='AIC')
    p_value = result[1]

    if p_value < best_p_value:
        best_p_value = p_value
        best_d = d

print(f"\nBest d value found: {best_d} (p-value = {best_p_value:.4f})")
```

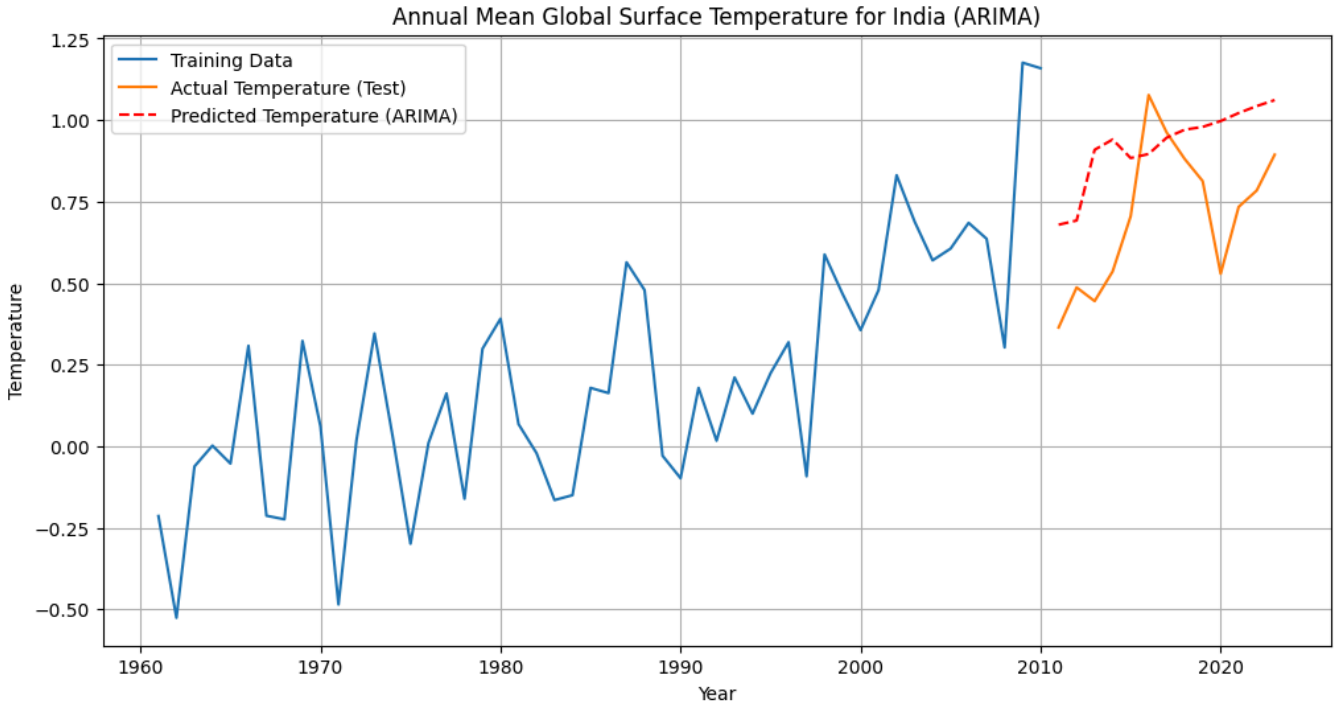
```
Best d value found: 1 (p-value = 0.0000)
```

```
!pip install pmdarima
from pmdarima import auto_arima

# Fit Auto-ARIMA
stepwise_fit = auto_arima(temperatures, trace=True, suppress_warnings=True)

# Print the best parameters and AIC
print(stepwise_fit.summary())
```

```
Collecting pmdarima
  Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl.metadata (7.8 kB)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.4.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (3.0.1
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.26.4)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.2.2)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.13.1)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.2.3)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (75.1.0)
Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (24.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2024.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->pmdarima
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (1.0.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=0.19->pm
Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (2.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 36.9 MB/s eta 0:00:00
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.4
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.37 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=34.433, Time=0.03 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=32.350, Time=0.03 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.17 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=32.640, Time=0.01 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=15.014, Time=0.07 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=16.852, Time=0.06 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.27 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.24 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.38 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=14.030, Time=0.11 sec
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=30.723, Time=0.02 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=15.942, Time=0.08 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=12.772, Time=0.32 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=16.582, Time=0.12 sec
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=13.417, Time=0.76 sec
 ARIMA(2,1,2)(0,0,0)[0]             : AIC=inf, Time=2.16 sec
 ARIMA(1,1,2)(0,0,0)[0]             : AIC=inf, Time=0.72 sec
 ARIMA(3,1,2)(0,0,0)[0]             : AIC=inf, Time=1.45 sec

Best model:  ARIMA(2,1,1)(0,0,0)[0]
Total fit time: 7.416 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   63
Model:               SARIMAX(2, 1, 1)   Log Likelihood                -2.386
Date:                Sat, 14 Dec 2024   AIC                           12.772
Time:                        02:12:19   BIC                           21.281
Sample:                             0   HQIC                          16.113
                                 - 63
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
```

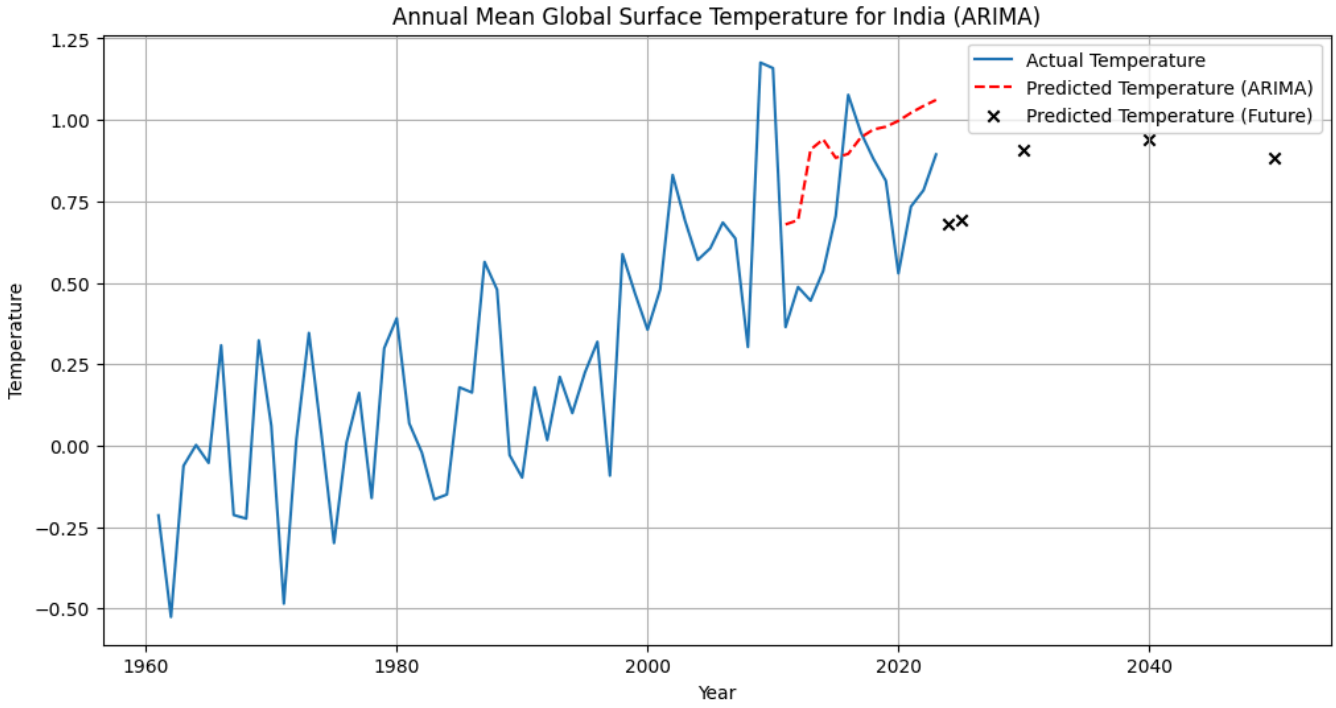
```
!pip install pmdarima

import pmdarima as pm
from pmdarima import model_selection
import numpy as np
import matplotlib.pyplot as plt


# Split the data into training and testing sets
train_data, test_data = model_selection.train_test_split(temperatures, train_size=0.8)

# Fit an ARIMA model
arima_model = pm.ARIMA(order=(2,1,1))
arima_model.fit(train_data)

# Make predictions
n_periods = len(test_data)
predictions = arima_model.predict(n_periods=n_periods)


# Plot the results
plt.figure(figsize=(12, 6))
plt.plot(years[:len(train_data)], train_data, label='Training Data')
plt.plot(years[len(train_data):], test_data, label='Actual Temperature (Test)')
plt.plot(years[len(train_data):], predictions, label='Predicted Temperature (ARIMA)', linestyle='--', color='red')

plt.xlabel('Year')
plt.ylabel('Temperature')
plt.title('Annual Mean Global Surface Temperature for India (ARIMA)')
plt.legend()
plt.grid(True)
plt.show()


# Evaluate the model (example metrics)
from sklearn.metrics import mean_squared_error, mean_absolute_error

rmse = np.sqrt(mean_squared_error(test_data, predictions))
mae = mean_absolute_error(test_data, predictions)

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

```
Requirement already satisfied: pmdarima in /usr/local/lib/python3.10/dist-packages (2.0.4)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.4.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (3.0.11)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.26.4)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.2.2)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.13.1)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.2.3)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (75.1.0)
Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (24.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2.8
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2024.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->pmdarima)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (1.0.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=0.19->pmdar
```




```
Root Mean Squared Error (RMSE): 0.27978891549344265
Mean Absolute Error (MAE): 0.24630695281713907
```

```
# Forecasting future values
future_years = [2024, 2025, 2030, 2040, 2050]
future_predictions = arima_model.predict(n_periods=len(future_years))

plt.figure(figsize=(12, 6))
plt.plot(years, temperatures, label='Actual Temperature')
plt.plot(years[len(train_data):], predictions, label='Predicted Temperature (ARIMA)', linestyle='--', color='red')
plt.scatter(future_years, future_predictions, color='black', label='Predicted Temperature (Future)', marker='x')

plt.xlabel('Year')
plt.ylabel('Temperature')
plt.title('Annual Mean Global Surface Temperature for India (ARIMA)')
plt.legend()
plt.grid(True)
plt.show()

for year, temp in zip(future_years, future_predictions):
  print(f"Predicted temperature for {year}: {temp:.2f}")
```




```
Predicted temperature for 2024: 0.68
Predicted temperature for 2025: 0.69
Predicted temperature for 2030: 0.91
Predicted temperature for 2040: 0.94
Predicted temperature for 2050: 0.88
```

Pros:

Good for short-term forecasting

Only needs historical data

Models non-stationary data

Cons:

Not built for long-term forecasting

Poor at predicting turning points

Computationally expensive

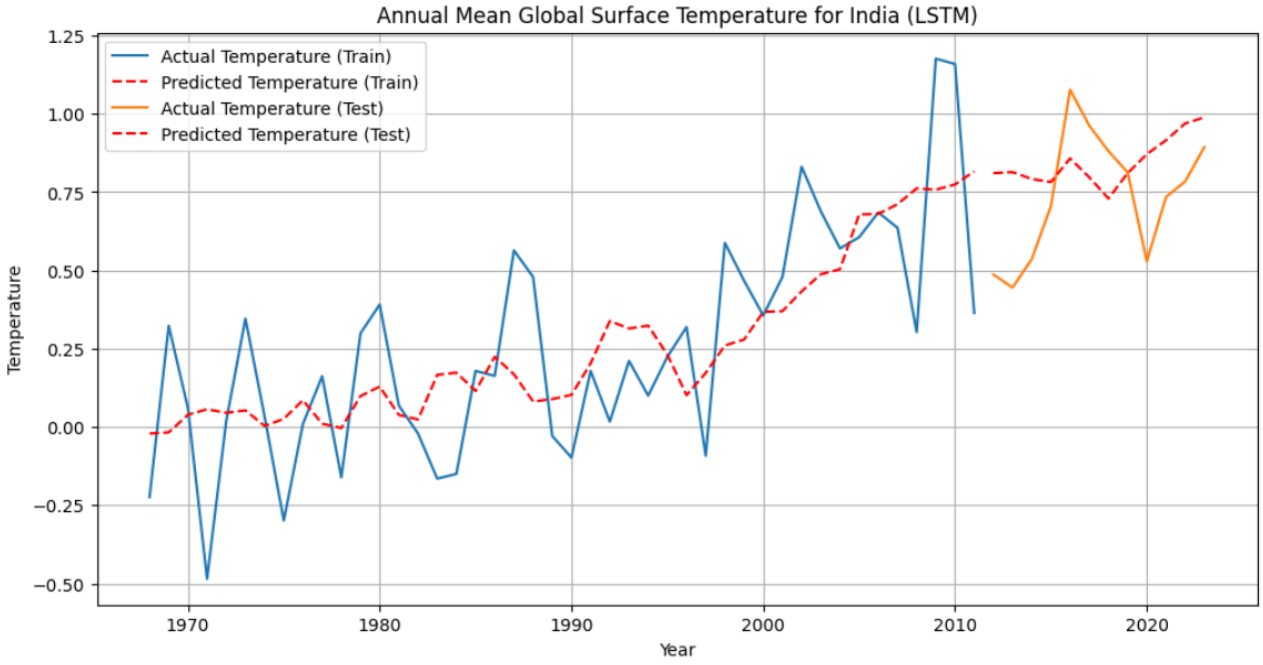
Parameters are subjective

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import numpy as np
import matplotlib.pyplot as plt
timesteps = 7
X, y = [], []
for i in range(len(temperatures) - timesteps):
    X.append(temperatures[i:(i + timesteps)])
    y.append(temperatures[i + timesteps])
X = np.array(X).reshape(-1, timesteps, 1)
y = np.array(y)


# Split the data into training and testing sets
split_ratio = 0.8
split_index = int(len(X) * split_ratio)
X_train, X_test = X[:split_index], X[split_index:]
y_train, y_test = y[:split_index], y[split_index:]

# Build the LSTM model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(timesteps, 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

# Train the model
model.fit(X_train, y_train, epochs=100, verbose=1)

# Make predictions
```

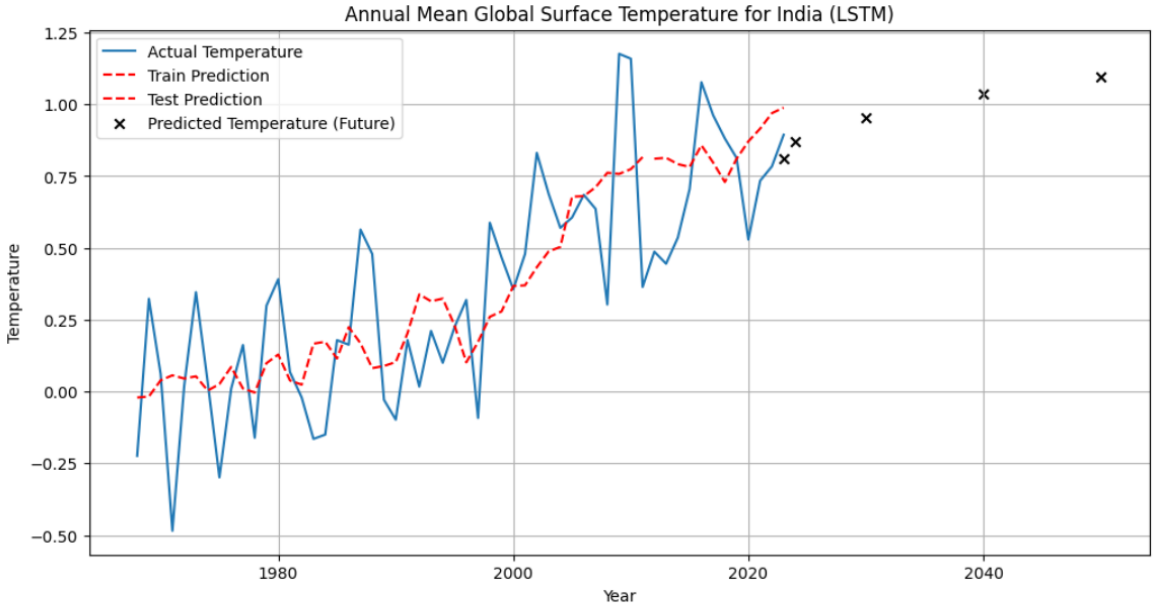
```
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0624
Epoch 74/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0648
Epoch 75/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0645
Epoch 76/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0644
Epoch 77/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0641
Epoch 78/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 0.0699
Epoch 79/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0693
Epoch 80/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0681
Epoch 81/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0652
Epoch 82/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0688
Epoch 83/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0669
Epoch 84/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0644
Epoch 85/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0652
Epoch 86/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0623
Epoch 87/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0680
Epoch 88/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0653
Epoch 89/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0671
Epoch 90/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0645
Epoch 91/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0638
Epoch 92/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0671
Epoch 93/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0653
Epoch 94/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0642
Epoch 95/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0626
Epoch 96/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0639
Epoch 97/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 0.0646
Epoch 98/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0644
Epoch 99/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 0.0649
Epoch 100/100
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 0.0633
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 144ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
```

```
# Plot the results
plt.figure(figsize=(12, 6))
plt.plot(years[timesteps:split_index+timesteps], y_train, label='Actual Temperature (Train)')
plt.plot(years[timesteps:split_index+timesteps], train_predict, label='Predicted Temperature (Train)', linestyle='--', color='red')
plt.plot(years[split_index+timesteps:], y_test, label='Actual Temperature (Test)')
plt.plot(years[split_index+timesteps:], test_predict, label='Predicted Temperature (Test)', linestyle='--', color='red')
plt.xlabel('Year')
plt.ylabel('Temperature')
plt.title('Annual Mean Global Surface Temperature for India (LSTM)')
plt.legend()
plt.grid(True)
plt.show()
```




```python
# Prepare future data for prediction
future_years = [2023, 2024, 2030, 2040, 2050]
future_X = []
last_sequence = X[-5]  # Use the last sequence from the training data
for year in future_years:
    # Predict the next temperature in the sequence
    next_temp_pred = model.predict(np.array([last_sequence]))[0][0]

    # Append the prediction to the sequence (and remove the oldest value)
    new_sequence = np.append(last_sequence[1:], next_temp_pred)
    future_X.append(next_temp_pred)
    last_sequence = new_sequence.reshape(timesteps, 1)


# Plot the results
plt.figure(figsize=(12, 6))
plt.plot(years[timesteps:], y, label='Actual Temperature')
plt.plot(years[timesteps:len(train_predict)+timesteps], train_predict, label='Train Prediction', linestyle='--', color='red')
plt.plot(years[len(train_predict)+timesteps:], test_predict, label='Test Prediction', linestyle='--', color='red')
plt.scatter(future_years, future_X, color='black', label='Predicted Temperature (Future)', marker='x')
plt.xlabel('Year')
plt.ylabel('Temperature')
plt.title('Annual Mean Global Surface Temperature for India (LSTM)')
plt.legend()
plt.grid(True)
plt.show()

for year, temp in zip(future_years, future_X):
    print(f"Predicted temperature for {year}: {temp:.2f}")
```




```
Predicted temperature for 2023: 0.81
Predicted temperature for 2024: 0.87
Predicted temperature for 2030: 0.95
Predicted temperature for 2040: 1.04
Predicted temperature for 2050: 1.10
```

```
[24] # Evaluate the LSTM model
     train_rmse = np.sqrt(mean_squared_error(y_train, train_predict))
     test_rmse = np.sqrt(mean_squared_error(y_test, test_predict))

     print(f'Train RMSE: {train_rmse}')
     print(f'Test RMSE: {test_rmse}')

     # Summarize the LSTM model
     print(model.summary())
```

```
Train RMSE: 0.2517992425910778
Test RMSE: 0.22429270695072184
Model: "sequential"
```

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 50)	10,400
dense (Dense)	(None, 1)	51

```
Total params: 31,355 (122.48 KB)
Trainable params: 10,451 (40.82 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 20,904 (81.66 KB)
None
```

Atmospheric Co2 concentration Dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd

# Assuming your dataset is in a CSV file named 'your_dataset.csv'
# Replace 'your_dataset.csv' with the actual filename and path
df = pd.read_csv('/content/drive/My Drive/Project sem1/Datasets/CO2.csv')

# Now you can work with the dataframe 'df'
print(df.head())
```

```
   ObjectId Country  ISO2 ISO3  \
0         1   World   NaN  WLD
1         2   World   NaN  WLD
2         3   World   NaN  WLD
3         4   World   NaN  WLD
4         5   World   NaN  WLD

                                        Indicator               Unit  \
0  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million
1  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million
2  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million
3  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million
4  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million

                                          Source CTS_Code  \
0  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA
1  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA
2  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA
3  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA
4  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA

                                  CTS_Name  \
0  Atmospheric Carbon Dioxide Concentrations
1  Atmospheric Carbon Dioxide Concentrations
2  Atmospheric Carbon Dioxide Concentrations
3  Atmospheric Carbon Dioxide Concentrations
4  Atmospheric Carbon Dioxide Concentrations

                               CTS_Full_Descriptor     Date   Value
0  Environment, Climate Change, Climate and Weath...  1958M03  315.70
1  Environment, Climate Change, Climate and Weath...  1958M04  317.45
2  Environment, Climate Change, Climate and Weath...  1958M05  317.51
3  Environment, Climate Change, Climate and Weath...  1958M06  317.24
4  Environment, Climate Change, Climate and Weath...  1958M07  315.86
```

```
df = df.drop('ISO2', axis=1)

print(df.head())
```

```
   ObjectId Country ISO3                                          Indicator  \
0         1   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations
1         2   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations
2         3   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations
3         4   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations
4         5   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations

                Unit                                             Source  \
0  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...
1  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...
2  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...
3  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...
4  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...

  CTS_Code                                  CTS_Name  \
0     ECCA  Atmospheric Carbon Dioxide Concentrations
1     ECCA  Atmospheric Carbon Dioxide Concentrations
2     ECCA  Atmospheric Carbon Dioxide Concentrations
3     ECCA  Atmospheric Carbon Dioxide Concentrations
4     ECCA  Atmospheric Carbon Dioxide Concentrations

                               CTS_Full_Descriptor     Date   Value
0  Environment, Climate Change, Climate and Weath...  1958M03  315.70
1  Environment, Climate Change, Climate and Weath...  1958M04  317.45
2  Environment, Climate Change, Climate and Weath...  1958M05  317.51
3  Environment, Climate Change, Climate and Weath...  1958M06  317.24
4  Environment, Climate Change, Climate and Weath...  1958M07  315.86
```

```
df.dtypes
```

	0
ObjectId	int64
Country	object
ISO3	object
Indicator	object
Unit	object
Source	object
CTS_Code	object
CTS_Name	object
CTS_Full_Descriptor	object
Date	object
Value	float64

dtype: object

```
# Replace 'M' with '-' in the 'Date' column
df['Date'] = df['Date'].str.replace('M', '-')

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

print(df.dtypes)
print(df.head())
```

```
ObjectId                     int64
Country                     object
ISO3                        object
Indicator                   object
Unit                        object
Source                      object
CTS_Code                    object
CTS_Name                    object
CTS_Full_Descriptor         object
Date                datetime64[ns]
Value                      float64
dtype: object
   ObjectId Country ISO3                                      Indicator  \
0         1   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations
1         2   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations
2         3   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations
3         4   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations
4         5   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations

                Unit                                            Source  \
0  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...
1  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...
2  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...
3  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...
4  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...

  CTS_Code                                  CTS_Name  \
0     ECCA  Atmospheric Carbon Dioxide Concentrations
1     ECCA  Atmospheric Carbon Dioxide Concentrations
2     ECCA  Atmospheric Carbon Dioxide Concentrations
3     ECCA  Atmospheric Carbon Dioxide Concentrations
4     ECCA  Atmospheric Carbon Dioxide Concentrations

                                CTS_Full_Descriptor       Date   Value
0  Environment, Climate Change, Climate and Weath... 1958-03-01  315.70
1  Environment, Climate Change, Climate and Weath... 1958-04-01  317.45
2  Environment, Climate Change, Climate and Weath... 1958-05-01  317.51
3  Environment, Climate Change, Climate and Weath... 1958-06-01  317.24
4  Environment, Climate Change, Climate and Weath... 1958-07-01  315.86
```

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

```
ObjectId              0
Country               0
ISO3                  0
Indicator             0
Unit                  0
Source                0
CTS_Code              0
CTS_Name              0
CTS_Full_Descriptor   0
Date                  0
```

```
Value                 0
dtype: int64
```

```
# Remove rows where the 'Unit' column contains 'Percent'
df = df[~df['Unit'].str.contains('Percent')]
print(df.head(5))
print(df.tail(5))
```

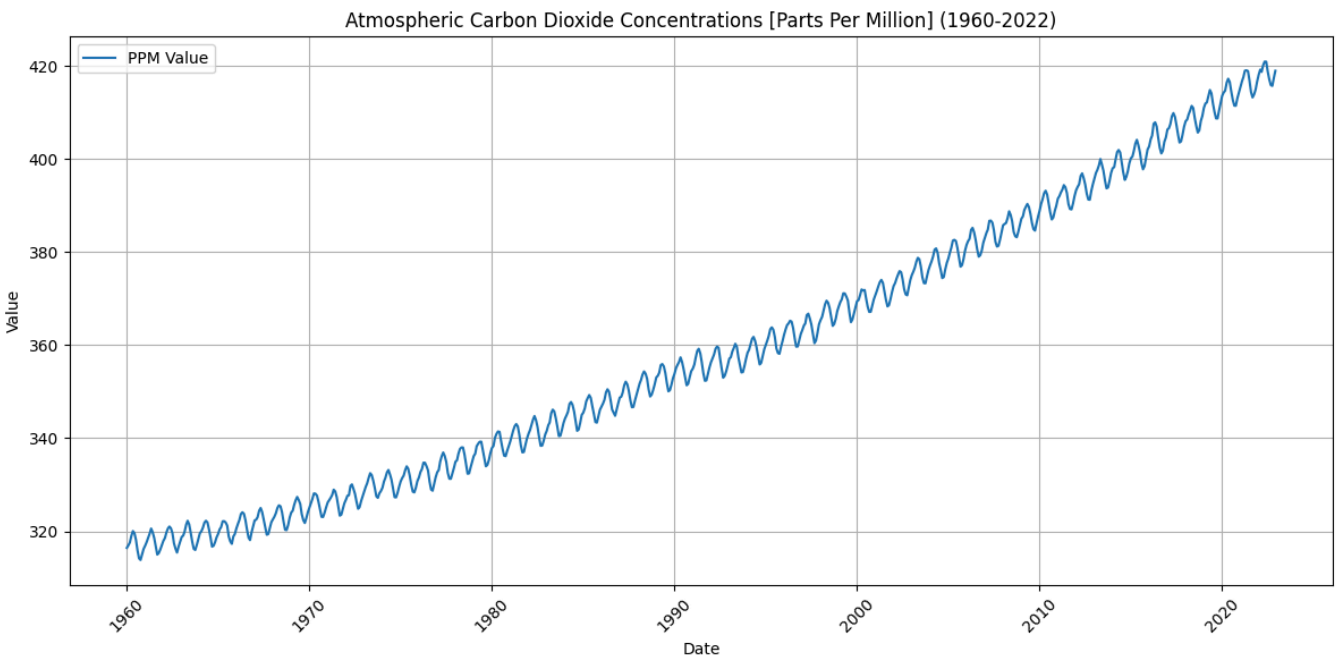
```
   ObjectId Country ISO3                                          Indicator  \
0         1   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations
1         2   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations
2         3   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations
3         4   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations
4         5   World  WLD  Monthly Atmospheric Carbon Dioxide Concentrations

                Unit                                             Source  \
0  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...
1  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...
2  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...
3  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...
4  Parts Per Million  Dr. Pieter Tans, National Oceanic and Atmosphe...

  CTS_Code                                CTS_Name  \
0     ECCA  Atmospheric Carbon Dioxide Concentrations
1     ECCA  Atmospheric Carbon Dioxide Concentrations
2     ECCA  Atmospheric Carbon Dioxide Concentrations
3     ECCA  Atmospheric Carbon Dioxide Concentrations
4     ECCA  Atmospheric Carbon Dioxide Concentrations

                                CTS_Full_Descriptor       Date   Value
0  Environment, Climate Change, Climate and Weath... 1958-03-01  315.70
1  Environment, Climate Change, Climate and Weath... 1958-04-01  317.45
2  Environment, Climate Change, Climate and Weath... 1958-05-01  317.51
3  Environment, Climate Change, Climate and Weath... 1958-06-01  317.24
4  Environment, Climate Change, Climate and Weath... 1958-07-01  315.86
      ObjectId Country ISO3  \
1560      1561   World  WLD
1562      1563   World  WLD
1564      1565   World  WLD
1566      1567   World  WLD
1568      1569   World  WLD

                                            Indicator               Unit  \
1560  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million
1562  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million
1564  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million
1566  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million
1568  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million

                                              Source CTS_Code  \
1560  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA
1562  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA
1564  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA
1566  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA
1568  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA

                                     CTS_Name  \
1560  Atmospheric Carbon Dioxide Concentrations
1562  Atmospheric Carbon Dioxide Concentrations
1564  Atmospheric Carbon Dioxide Concentrations
1566  Atmospheric Carbon Dioxide Concentrations
1568  Atmospheric Carbon Dioxide Concentrations

                                  CTS_Full_Descriptor       Date   Value
1560  Environment, Climate Change, Climate and Weath... 2023-09-01  418.51
1562  Environment, Climate Change, Climate and Weath... 2023-10-01  418.82
```

```
df = df[:-1] # removing last 0.68 1568th row

print(df.tail(5))
```

```
      ObjectId Country ISO3  \
1558      1559   World  WLD
1560      1561   World  WLD
1562      1563   World  WLD
1564      1565   World  WLD
1566      1567   World  WLD

                                            Indicator               Unit  \
1558  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million
1560  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million
1562  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million
1564  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million
1566  Monthly Atmospheric Carbon Dioxide Concentrations  Parts Per Million

                                              Source CTS_Code  \
1558  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA
1560  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA
```

```
1562  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA
1564  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA
1566  Dr. Pieter Tans, National Oceanic and Atmosphe...     ECCA

                                      CTS_Name  \
1558  Atmospheric Carbon Dioxide Concentrations
1560  Atmospheric Carbon Dioxide Concentrations
1562  Atmospheric Carbon Dioxide Concentrations
1564  Atmospheric Carbon Dioxide Concentrations
1566  Atmospheric Carbon Dioxide Concentrations

                                   CTS_Full_Descriptor       Date   Value
1558  Environment, Climate Change, Climate and Weath... 2023-08-01  419.68
1560  Environment, Climate Change, Climate and Weath... 2023-09-01  418.51
1562  Environment, Climate Change, Climate and Weath... 2023-10-01  418.82
1564  Environment, Climate Change, Climate and Weath... 2023-11-01  420.46
1566  Environment, Climate Change, Climate and Weath... 2023-12-01  421.86
```

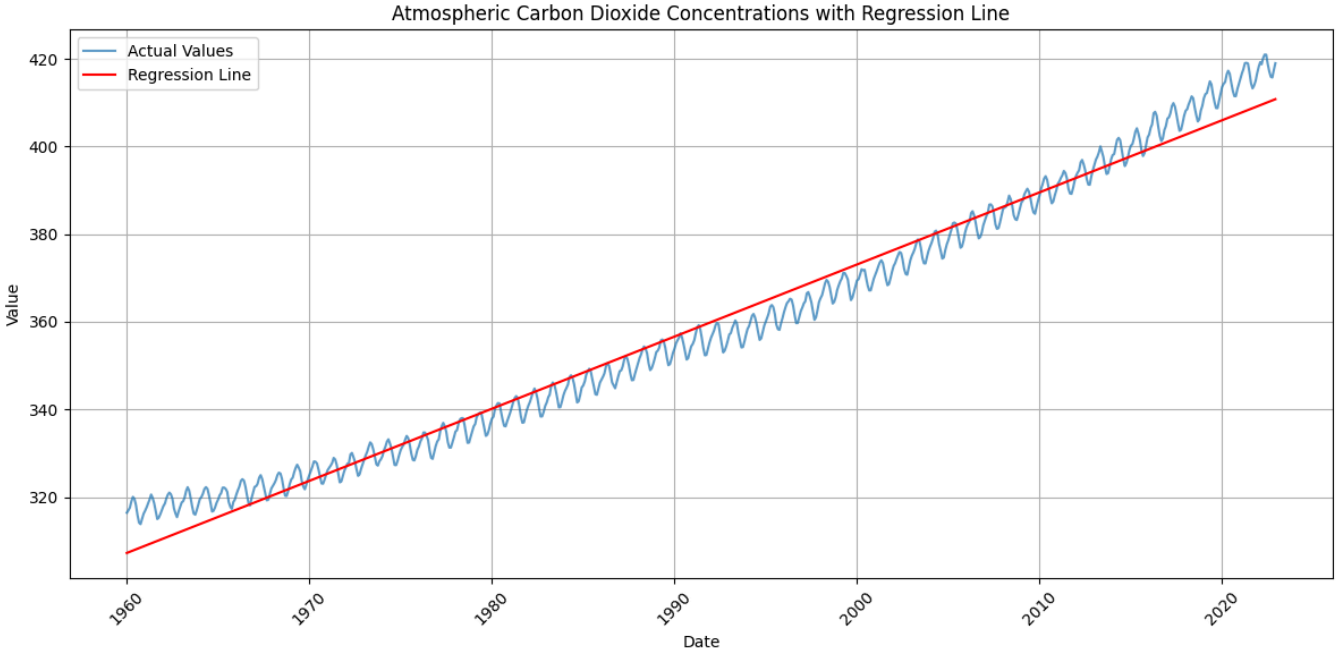
```
# Filter data for the specified period
df_filtered = df[(df['Date'] >= '1960-01-01') & (df['Date'] <= '2022-12-31')]
import matplotlib.pyplot as plt
import seaborn as sns
# Plot using seaborn
plt.figure(figsize=(12, 6))
sns.lineplot(x='Date', y='Value', data=df_filtered, label='PPM Value')

plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Atmospheric Carbon Dioxide Concentrations [Parts Per Million] (1960-2022)')
plt.grid(True)
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```




```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Prepare the data for regression
df_filtered['Time'] = (df_filtered['Date'] - df_filtered['Date'].min()).dt.days
X = df_filtered[['Time']]
y = df_filtered['Value']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the linear regression model
```

```
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Plot the regression line along with the original data
plt.figure(figsize=(12, 6))
sns.lineplot(x='Date', y='Value', data=df_filtered, label='Actual Values', alpha=0.7)
plt.plot(df_filtered['Date'], model.predict(X), color='red', label='Regression Line')

plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Atmospheric Carbon Dioxide Concentrations with Regression Line')
plt.grid(True)
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```

```
<ipython-input-10-45ef66755554>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df_filtered['Time'] = (df_filtered['Date'] - df_filtered['Date'].min()).dt.days
```

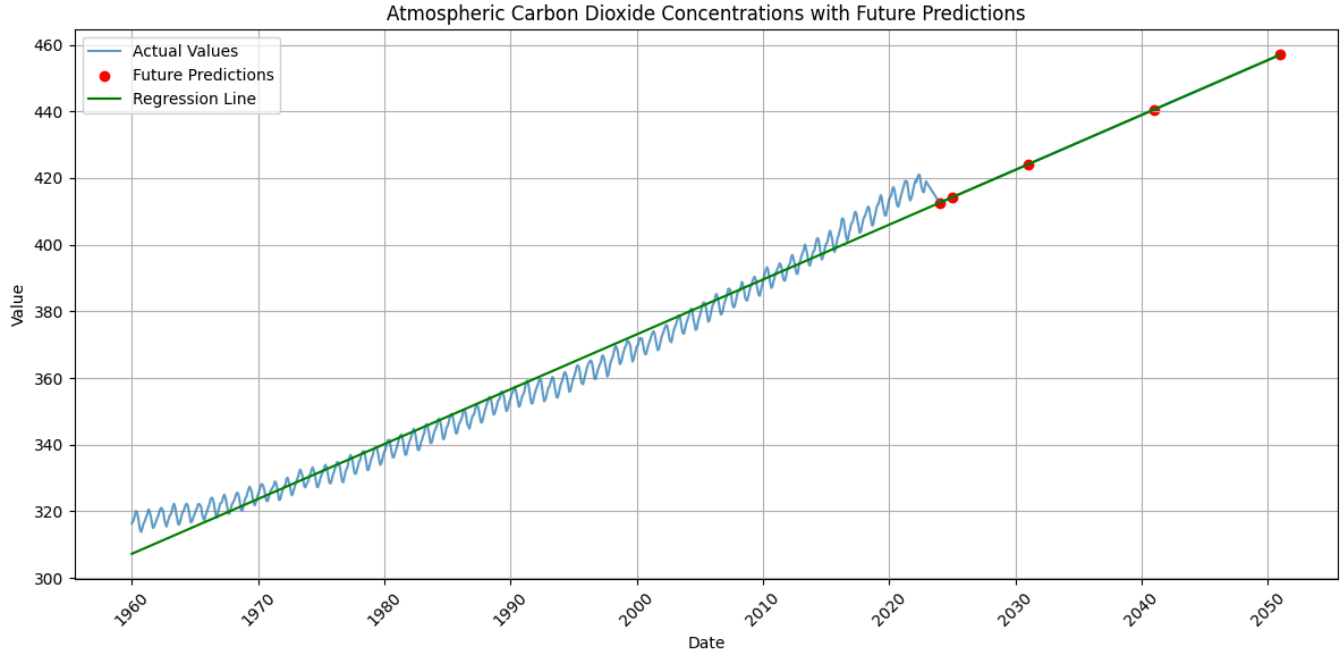



```
# Print the model's coefficients and intercept
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)

# Evaluate the model (example using R-squared)
from sklearn.metrics import r2_score
r2 = r2_score(y_test, y_pred)*100
print(f"R-squared: {r2} %")

# You can add other evaluation metrics as needed, such as Mean Squared Error (MSE) or Mean Absolute Error (MAE)
from sklearn.metrics import mean_squared_error, mean_absolute_error
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
```

```
Coefficients: [0.00450561]
Intercept: 307.2485750703215
R-squared: 97.74697051786634 %
Mean Squared Error: 19.107936506686393
Mean Absolute Error: 3.6588814342312617
```

```
# Predict values for future years
future_dates = pd.to_datetime(['2023-12-31', '2024-12-31', '2030-12-31','2040-12-31', '2050-12-31'])
# Changed the following line to get days attribute from the TimedeltaIndex
future_time = (future_dates - df_filtered['Date'].min()).days
future_predictions = model.predict(future_time.values.reshape(-1, 1))

# Create a DataFrame for future predictions
future_df = pd.DataFrame({'Date': future_dates, 'Value': future_predictions})

# Concatenate the original data and future predictions
combined_df = pd.concat([df_filtered, future_df], ignore_index=True)

# Plot the combined data
plt.figure(figsize=(12, 6))
sns.lineplot(x='Date', y='Value', data=combined_df, label='Actual Values', alpha=0.7)

# Highlight the future predictions
plt.scatter(future_df['Date'], future_df['Value'], color='red', label='Future Predictions')

# Note: Ensure 'Time' column is available in combined_df, or adjust accordingly
plt.plot(combined_df['Date'], model.predict((combined_df['Date'] - combined_df['Date'].min()).dt.days.values.reshape(-1,1)), color='gree

plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Atmospheric Carbon Dioxide Concentrations with Future Predictions')
plt.grid(True)
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
```

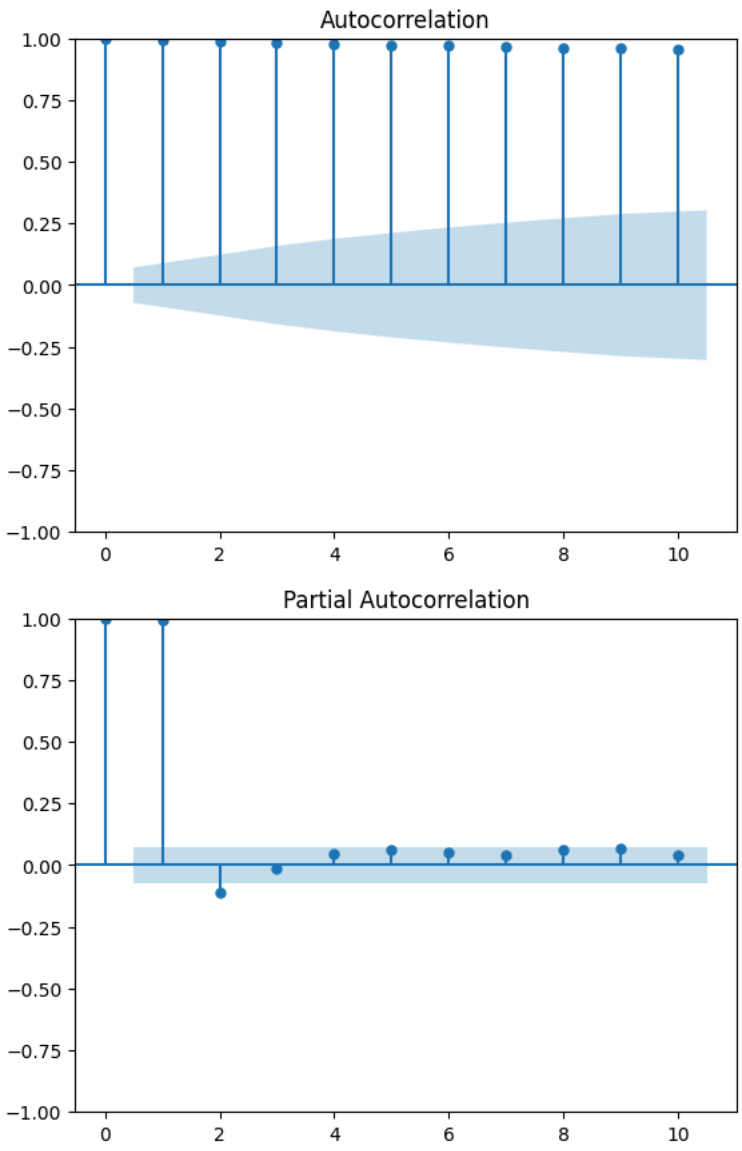



```
# Predict values for future years
future_dates = pd.to_datetime(['2023-12-31', '2024-12-31', '2030-12-31','2040-12-31', '2050-12-31'])
future_time = (future_dates - df_filtered['Date'].min()).days
future_predictions = model.predict(future_time.values.reshape(-1, 1))

print("Future Predictions:")
for date, prediction in zip(future_dates, future_predictions):
    print(f"Date: {date}, Predicted Value: {prediction}")

# Create a DataFrame for future predictions
future_df = pd.DataFrame({'Date': future_dates, 'Value': future_predictions})
```

```
Future Predictions:
Date: 2023-12-31 00:00:00, Predicted Value: 412.5672139719967
```

```
Date: 2024-12-31 00:00:00, Predicted Value: 414.2162673126603
Date: 2030-12-31 00:00:00, Predicted Value: 424.08805930554036
Date: 2040-12-31 00:00:00, Predicted Value: 440.5470534406342
Date: 2050-12-31 00:00:00, Predicted Value: 457.0015419655077
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:493: UserWarning: X does not have valid feature names, but LinearRegression
  warnings.warn(
```

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Calculate ACF and PACF
plot_acf(df_filtered['Value'], lags=10)  # Adjust lags as needed
plt.show()

plot_pacf(df_filtered['Value'], lags=10)  # Adjust lags as needed
plt.show()
```




```
from statsmodels.tsa.stattools import adfuller

# Perform the Augmented Dickey-Fuller test
result = adfuller(df_filtered['Value'])

# Print the test results
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

# Interpret the results
p_value = result[1]
if p_value <= 0.05:
    print("The series is stationary (reject the null hypothesis)")
else:
```

```
    print("The series is non-stationary (fail to reject the null hypothesis)")

p = p_value
q = result[0] #ADF Statistic

print(f"p-value: {p}")
print(f"ADF Statistic: {q}")
```

```
ADF Statistic: 5.075071
p-value: 1.000000
Critical Values:
        1%: -3.439
        5%: -2.865
        10%: -2.569
The series is non-stationary (fail to reject the null hypothesis)
p-value: 1.0
ADF Statistic: 5.075071484750594
```

```
import numpy as np
# Perform the Augmented Dickey-Fuller test
result = adfuller(df_filtered['Value'])
p_value = result[1]
adf_statistic = result[0]

print(f"Initial p-value: {p_value}")
print(f"Initial ADF Statistic: {adf_statistic}")

def adjust_data(data, d):
    #Example: differencing the data 'd' times
    adjusted_data = data.copy()
    for _ in range(d):
      adjusted_data = adjusted_data.diff().dropna()
    return adjusted_data

#Testing various differencing values
for d in range(1, 4):  # Test d values from 1 to 3 (inclusive)
  adjusted_series = adjust_data(df_filtered['Value'],d)

  if len(adjusted_series)>0: #Check if the adjusted series is not empty after differencing.
      result_adjusted = adfuller(adjusted_series)
      p_value_adjusted = result_adjusted[1]
      adf_statistic_adjusted = result_adjusted[0]

      print(f"\nDifferencing order (d): {d}")
      print(f"Adjusted p-value: {p_value_adjusted}")
      print(f"Adjusted ADF Statistic: {adf_statistic_adjusted}")

      if p_value_adjusted <= 0.05:
          print(f"For d = {d}, the series is likely stationary.")
          break # Stop if stationary is found.
  else:
    print(f"\nDifferencing order (d): {d} leads to an empty series.")
```

```
Initial p-value: 1.0
Initial ADF Statistic: 5.075071484750594

Differencing order (d): 1
Adjusted p-value: 1.0477685570090106e-05
Adjusted ADF Statistic: -5.1625842162862385
For d = 1, the series is likely stationary.
```

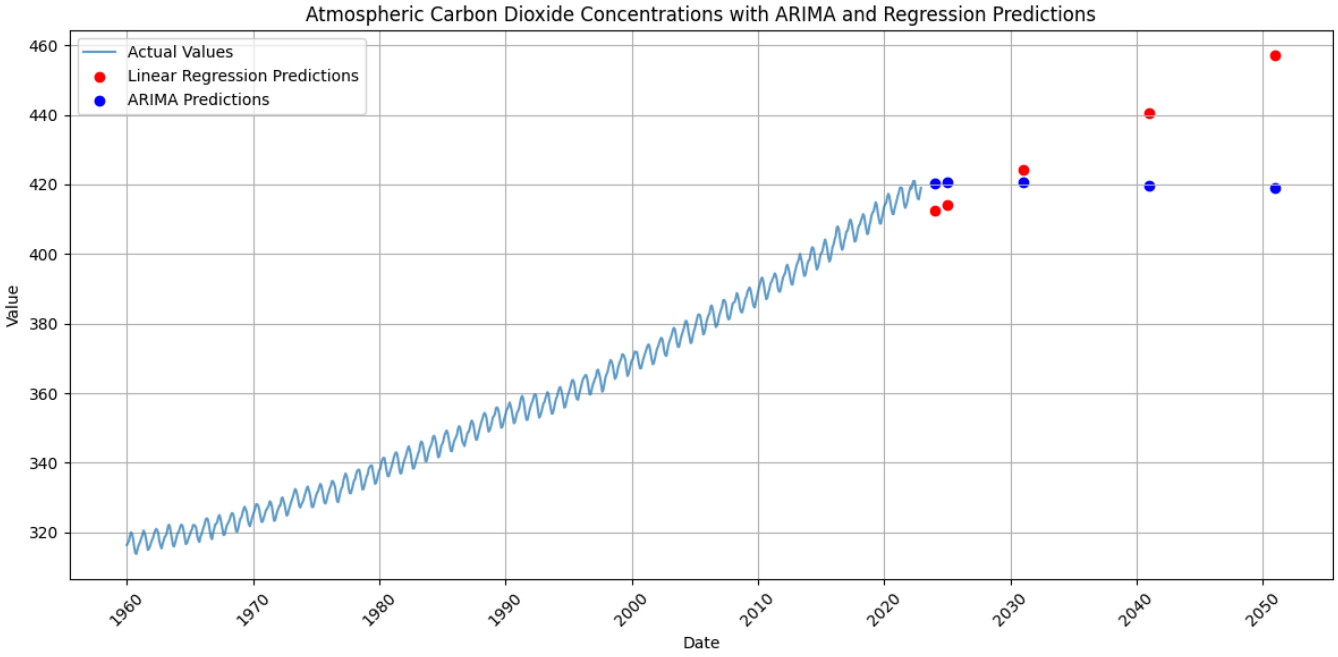
```
!pip install pmdarima
from pmdarima import auto_arima

# Assuming 'df_filtered' and 'y' are defined as in the previous code
# Fit Auto-ARIMA
stepwise_fit = auto_arima(y, trace=True, suppress_warnings=True)

# Print the best model's parameters and AIC
print(stepwise_fit.summary())
```

```
Collecting pmdarima
  Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl.metadata (7.8 kB)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.4.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (3.0.1
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.26.4)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.2.2)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.13.1)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.2.3)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (75.1.0)
Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (24.2)
```

```
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2024.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->pmdarima
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (1.0.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=0.19->pm
Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (2.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 13.2 MB/s eta 0:00:00
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.4
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=1510.855, Time=1.66 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=2469.183, Time=0.04 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=1961.209, Time=0.11 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=2019.580, Time=0.23 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=2476.216, Time=0.04 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=1791.916, Time=0.31 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=1509.595, Time=0.59 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=1860.172, Time=0.24 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=1755.361, Time=0.16 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=1510.991, Time=0.93 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=1681.833, Time=0.25 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=1511.946, Time=1.73 sec
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=1606.290, Time=0.14 sec

Best model:  ARIMA(2,1,1)(0,0,0)[0] intercept
Total fit time: 6.424 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  756
Model:               SARIMAX(2, 1, 1)   Log Likelihood                -749.798
Date:                Thu, 12 Dec 2024   AIC                           1509.595
Time:                        07:05:40   BIC                           1532.729
Sample:                             0   HQIC                          1518.506
                                  - 756
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0412      0.003     11.886      0.000       0.034       0.048
ar.L1          1.5441      0.022     70.329      0.000       1.501       1.587
ar.L2         -0.8465      0.023    -37.316      0.000      -0.891      -0.802
ma.L1         -0.8989      0.018    -49.566      0.000      -0.934      -0.863
sigma2         0.4251      0.022     19.561      0.000       0.382       0.468
```

```
from statsmodels.tsa.arima.model import ARIMA
import numpy as np

p = 2
d = 1
q = 1


# Fit the ARIMA model
model = ARIMA(df_filtered['Value'], order=(p, d, q))
model_fit = model.fit()


# Make predictions for future dates
future_predictions_arima = model_fit.predict(start=len(df_filtered), end=len(df_filtered) + len(future_dates) - 1)

# Create a DataFrame for future ARIMA predictions
future_df_arima = pd.DataFrame({'Date': future_dates, 'Value': future_predictions_arima})


print("Future Predictions using ARIMA:")
for date, prediction in zip(future_dates, future_predictions_arima):
    print(f"Date: {date}, Predicted Value: {prediction}")

# Plot the combined data with ARIMA predictions
plt.figure(figsize=(12, 6))
sns.lineplot(x='Date', y='Value', data=df_filtered, label='Actual Values', alpha=0.7)
plt.scatter(future_df['Date'], future_df['Value'], color='red', label='Linear Regression Predictions')
plt.scatter(future_df_arima['Date'], future_df_arima['Value'], color='blue', label='ARIMA Predictions')
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Atmospheric Carbon Dioxide Concentrations with ARIMA and Regression Predictions')
plt.grid(True)
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported index was provided. As a
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported index was provided. As a
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported index was provided. As a
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:837: ValueWarning: No supported index is available. Predic
  return get_prediction_index(
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:837: FutureWarning: No supported index is available. In th
  return get_prediction_index(
Future Predictions using ARIMA:
Date: 2023-12-31 00:00:00, Predicted Value: 420.1446884419008
Date: 2024-12-31 00:00:00, Predicted Value: 420.6556712411023
Date: 2030-12-31 00:00:00, Predicted Value: 420.48374825519846
Date: 2040-12-31 00:00:00, Predicted Value: 419.7988454326123
Date: 2050-12-31 00:00:00, Predicted Value: 418.8932374688619
```




```
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Normalize the 'Value' column
scaler = MinMaxScaler()
df_filtered['Value_scaled'] = scaler.fit_transform(df_filtered['Value'].values.reshape(-1, 1))

# Prepare the data for LSTM
def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
        X.append(a)
        Y.append(dataset[i + look_back, 0])
    return np.array(X), np.array(Y)

look_back = 10  # Adjust this value as needed
X, y = create_dataset(df_filtered['Value_scaled'].values.reshape(-1, 1), look_back)

# Reshape X for LSTM input
X = np.reshape(X, (X.shape[0], X.shape[1], 1))

# Split data into train and test sets
train_size = int(len(X) * 0.8)
test_size = len(X) - train_size
X_train, X_test = X[0:train_size,:], X[train_size:len(X),:]
y_train, y_test = y[0:train_size], y[train_size:len(y)]

# Create and train the LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50))
```

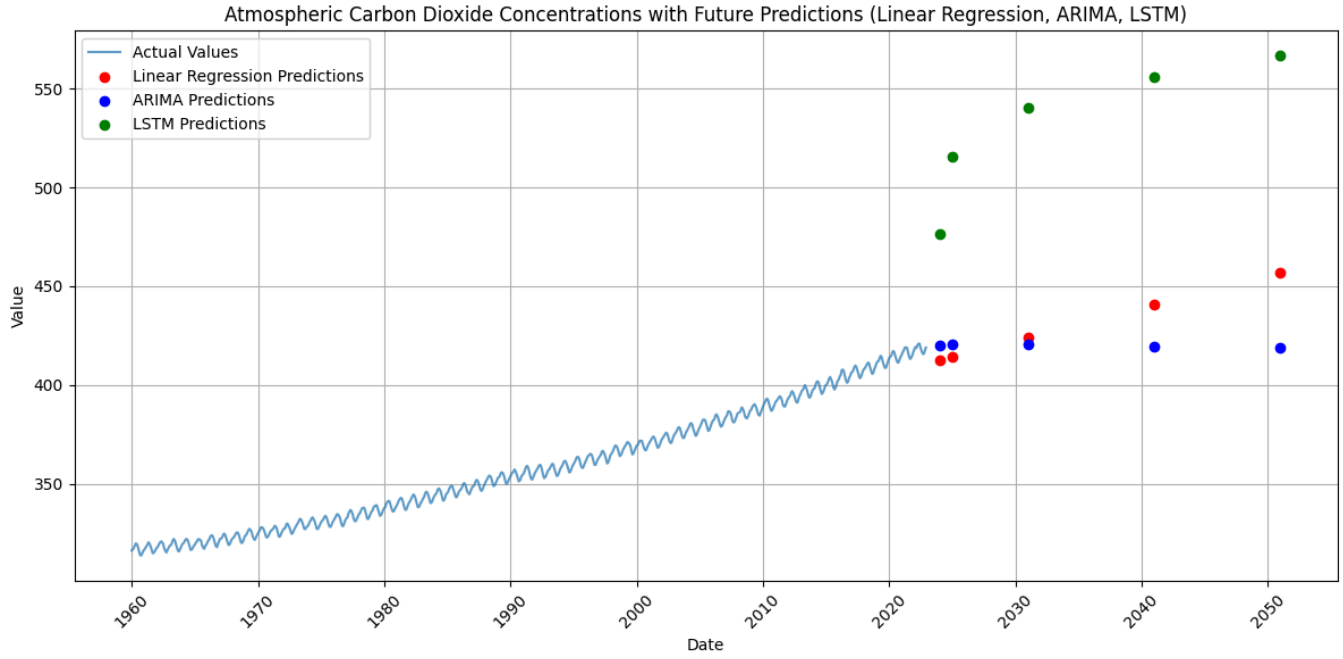
```
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=1) # Adjust epochs and batch_size

# Make predictions
train_predict = model.predict(X_train)
test_predict = model.predict(X_test)

# Inverse transform the predictions
train_predict = scaler.inverse_transform(train_predict)
y_train = scaler.inverse_transform([y_train])
test_predict = scaler.inverse_transform(test_predict)
y_test = scaler.inverse_transform([y_test])


# Plot the predictions
plt.figure(figsize=(12, 6))
plt.plot(scaler.inverse_transform(df_filtered['Value_scaled'].values.reshape(-1,1)), label='Actual')
plt.plot(np.concatenate((train_predict, test_predict)), label='Predicted')
plt.xlabel('Time')
plt.ylabel('CO2 Value')
plt.title('LSTM Predictions for CO2')
plt.legend()
plt.show()
```

```
<ipython-input-22-cae99dfa131d>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-vers
  df_filtered['Value_scaled'] = scaler.fit_transform(df_filtered['Value'].values.reshape(-1, 1))
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim` ar
  super().__init__(**kwargs)
Epoch 1/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 6s 16ms/step - loss: 0.0578
Epoch 2/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - loss: 0.0045
Epoch 3/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0011
Epoch 4/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 6.7726e-04
Epoch 5/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - loss: 7.0704e-04
Epoch 6/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step - loss: 6.5038e-04
Epoch 7/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 6.7625e-04
Epoch 8/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 6.6186e-04
Epoch 9/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 7.3815e-04
Epoch 10/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 6.8707e-04
Epoch 11/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 6.4444e-04
Epoch 12/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 7.3039e-04
Epoch 13/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 7.4004e-04
Epoch 14/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 6.5039e-04
Epoch 15/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - loss: 6.5172e-04
Epoch 16/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 6.3937e-04
Epoch 17/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 6.6003e-04
Epoch 18/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 6.4712e-04
Epoch 19/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - loss: 6.5149e-04
Epoch 20/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 6.1914e-04
Epoch 21/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 6.7000e-04
Epoch 22/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 6.4874e-04
Epoch 23/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 6.8052e-04
Epoch 24/50
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 7.3382e-04
Epoch 25/50
```

```
future_dates = pd.to_datetime(['2023-12-31', '2024-12-31', '2030-12-31', '2040-12-31', '2050-12-31'])

future_time = (future_dates - df_filtered['Date'].min()).days
future_time_scaled = scaler.transform(future_time.values.reshape(-1, 1))
```

```
# Create sequences for future predictions
future_X = []
last_sequence = X[-1]

for i in range(len(future_dates)):
  last_sequence = np.append(last_sequence[1:],future_time_scaled[i])
  future_X.append(last_sequence)

future_X = np.array(future_X).reshape(len(future_dates), look_back, 1)


# Predict future values using LSTM
future_predictions_lstm = model.predict(future_X)

# Inverse transform to get actual values
future_predictions_lstm = scaler.inverse_transform(future_predictions_lstm)


# Create DataFrame for LSTM future predictions
future_df_lstm = pd.DataFrame({'Date': future_dates, 'Value': future_predictions_lstm.flatten()})


# Plot all predictions together
plt.figure(figsize=(12, 6))
sns.lineplot(x='Date', y='Value', data=df_filtered, label='Actual Values', alpha=0.7)
plt.scatter(future_df['Date'], future_df['Value'], color='red', label='Linear Regression Predictions')
plt.scatter(future_df_arima['Date'], future_df_arima['Value'], color='blue', label='ARIMA Predictions')
plt.scatter(future_df_lstm['Date'], future_df_lstm['Value'], color='green', label='LSTM Predictions')
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Atmospheric Carbon Dioxide Concentrations with Future Predictions (Linear Regression, ARIMA, LSTM)')
plt.grid(True)
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
```




```
# Print the model summary LSTM
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 10, 50)	10,400
lstm_1 (LSTM)	(None, 50)	20,200
dense (Dense)	(None, 1)	51

Total params: 91,955 (359.20 KB)
Trainable params: 30,651 (119.73 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 61,304 (239.47 KB)

Change in Mean Sea Level Dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd


df = pd.read_csv('/content/drive/My Drive/Project sem1/Datasets/Change_in_Mean_Sea_Levels.csv')

# Now you can work with the dataframe 'df'
print(df.head())
```

```
   ObjectId Country  ISO2 ISO3  \
0         1   World   NaN  WLD
1         2   World   NaN  WLD
2         3   World   NaN  WLD
3         4   World   NaN  WLD
4         5   World   NaN  WLD

                                           Indicator         Unit  \
0  Change in mean sea level: Sea level: TOPEX.Pos...  Millimeters
1  Change in mean sea level: Sea level: TOPEX.Pos...  Millimeters
2  Change in mean sea level: Sea level: TOPEX.Pos...  Millimeters
3  Change in mean sea level: Sea level: TOPEX.Pos...  Millimeters
4  Change in mean sea level: Sea level: TOPEX.Pos...  Millimeters

                                          Source CTS_Code  \
0  National Oceanic and Atmospheric Administratio...     ECCL
1  National Oceanic and Atmospheric Administratio...     ECCL
2  National Oceanic and Atmospheric Administratio...     ECCL
3  National Oceanic and Atmospheric Administratio...     ECCL
4  National Oceanic and Atmospheric Administratio...     ECCL

                   CTS_Name  \
0  Change in Mean Sea Level
1  Change in Mean Sea Level
2  Change in Mean Sea Level
3  Change in Mean Sea Level
4  Change in Mean Sea Level

                               CTS_Full_Descriptor         Measure  \
0  Environment, Climate Change, Climate Indicator...     Andaman Sea
1  Environment, Climate Change, Climate Indicator...     Arabian Sea
2  Environment, Climate Change, Climate Indicator...  Atlantic Ocean
3  Environment, Climate Change, Climate Indicator...      Baltic Sea
4  Environment, Climate Change, Climate Indicator...      Bay Bengal

          Date   Value
0  D12/17/1992  -10.34
1  D12/17/1992  -18.46
2  D12/17/1992  -15.41
3  D12/17/1992  196.85
4  D12/17/1992    3.27
```

```
df = df.drop('ISO2', axis=1)

print(df.head())
```

```
   ObjectId Country ISO3                                          Indicator  \
0         1   World  WLD  Change in mean sea level: Sea level: TOPEX.Pos...
1         2   World  WLD  Change in mean sea level: Sea level: TOPEX.Pos...
2         3   World  WLD  Change in mean sea level: Sea level: TOPEX.Pos...
3         4   World  WLD  Change in mean sea level: Sea level: TOPEX.Pos...
4         5   World  WLD  Change in mean sea level: Sea level: TOPEX.Pos...

          Unit                                          Source CTS_Code  \
0  Millimeters  National Oceanic and Atmospheric Administratio...     ECCL
1  Millimeters  National Oceanic and Atmospheric Administratio...     ECCL
2  Millimeters  National Oceanic and Atmospheric Administratio...     ECCL
3  Millimeters  National Oceanic and Atmospheric Administratio...     ECCL
4  Millimeters  National Oceanic and Atmospheric Administratio...     ECCL

                   CTS_Name  \
0  Change in Mean Sea Level
1  Change in Mean Sea Level
2  Change in Mean Sea Level
3  Change in Mean Sea Level
4  Change in Mean Sea Level

                               CTS_Full_Descriptor         Measure  \
0  Environment, Climate Change, Climate Indicator...     Andaman Sea
1  Environment, Climate Change, Climate Indicator...     Arabian Sea
2  Environment, Climate Change, Climate Indicator...  Atlantic Ocean
3  Environment, Climate Change, Climate Indicator...      Baltic Sea
4  Environment, Climate Change, Climate Indicator...      Bay Bengal
```

```
          Date   Value
0  D12/17/1992  -10.34
1  D12/17/1992  -18.46
2  D12/17/1992  -15.41
3  D12/17/1992  196.85
4  D12/17/1992    3.27
```

```
df.dtypes
```

	0
ObjectId	int64
Country	object
ISO3	object
Indicator	object
Unit	object
Source	object
CTS_Code	object
CTS_Name	object
CTS_Full_Descriptor	object
Measure	object
Date	object
Value	float64

dtype: object

```
df['Date'] = pd.to_datetime(df['Date'].str.lstrip('D'), format='%m/%d/%Y', errors='coerce')

# Verify the updated data type
print(df.dtypes)
```

```
ObjectId                      int64
Country                      object
ISO3                         object
Indicator                    object
Unit                         object
Source                       object
CTS_Code                     object
CTS_Name                     object
CTS_Full_Descriptor          object
Measure                      object
Date                 datetime64[ns]
Value                       float64
dtype: object
```

```
print(df['Date'])
```

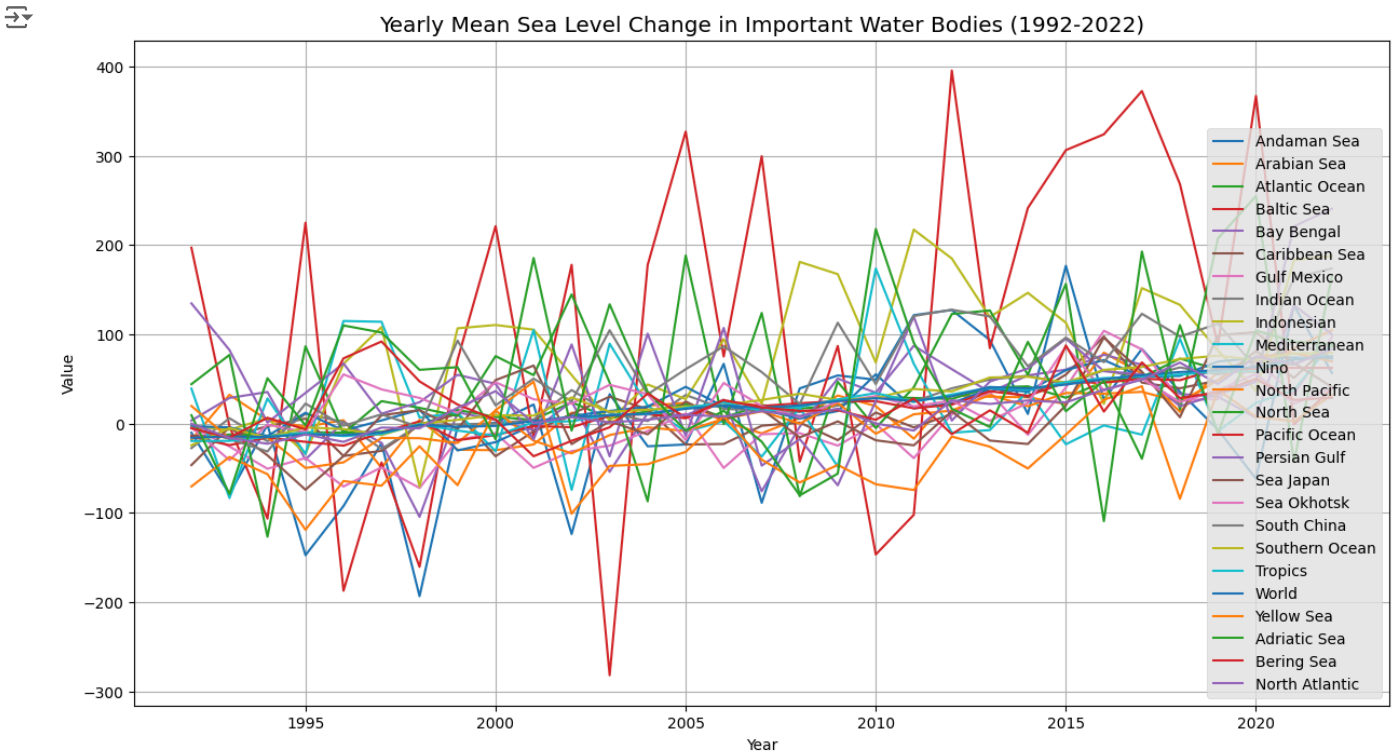
```
0       1992-12-17
1       1992-12-17
2       1992-12-17
3       1992-12-17
4       1992-12-17
           ...
35599   2022-11-08
35600   2022-11-08
35601   2022-11-08
35602   2022-11-08
35603   2022-11-08
Name: Date, Length: 35604, dtype: datetime64[ns]
```

```
unique_measures = df['Measure'].unique()
unique_measures
```

```
array(['Andaman Sea', 'Arabian Sea', 'Atlantic Ocean', 'Baltic Sea',
       'Bay Bengal', 'Caribbean Sea', 'Gulf Mexico', 'Indian Ocean',
       'Indonesian', 'Mediterranean', 'Nino', 'North Pacific',
       'North Sea', 'Pacific Ocean', 'Persian Gulf', 'Sea Japan',
       'Sea Okhotsk', 'South China', 'Southern Ocean', 'Tropics', 'World',
       'Yellow Sea', 'Adriatic Sea', 'Bering Sea', 'North Atlantic'],
      dtype=object)
```

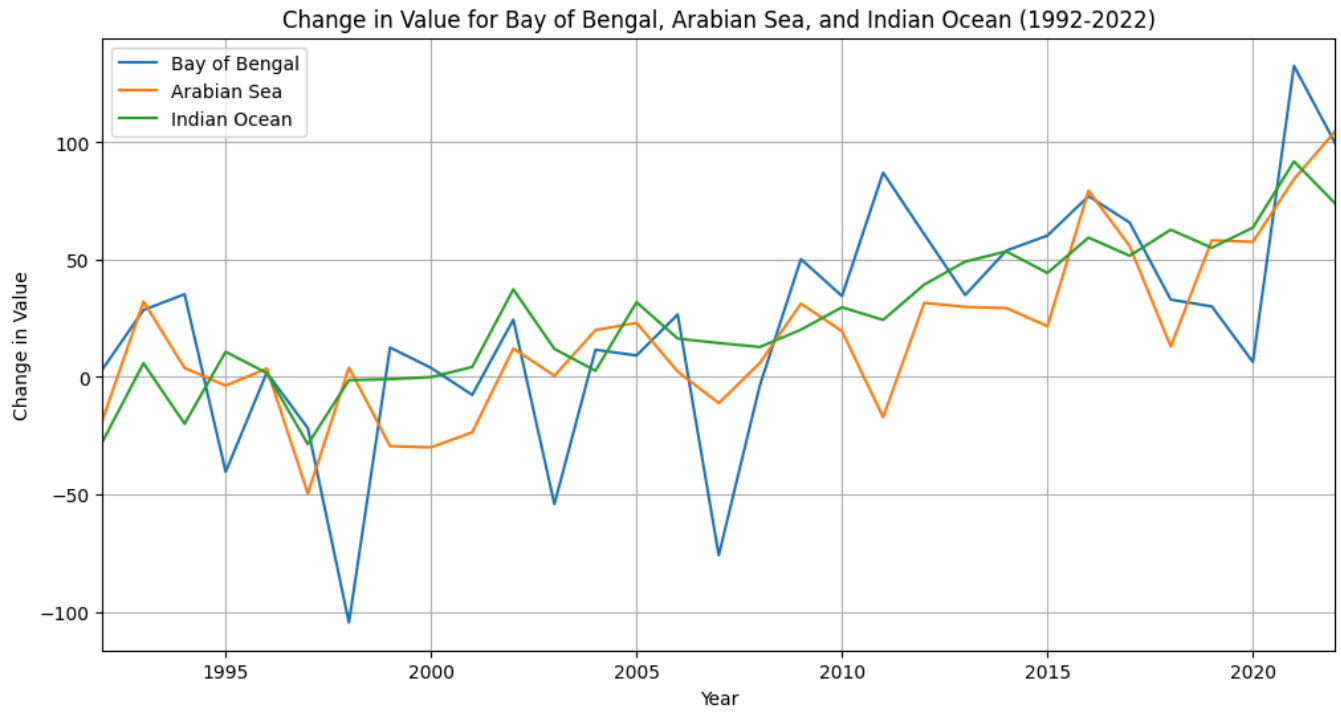
```
import matplotlib.pyplot as plt

plt.style.use('default')
```

```
# Create an empty dictionary to store the yearly changes
yearly_changes = {}

# Iterate over unique measures
for measure in unique_measures:
    yearly_changes[measure] = {}
    # Filter data for the current measure
    measure_df = df[df['Measure'] == measure]

    # Iterate through years from 1992 to 2022
    for year in range(1992, 2023):
        # Filter data for the current year
        year_df = measure_df[measure_df['Date'].dt.year == year]

        # Check if data exists for the year. If not, add NaN.
        if not year_df.empty:
           yearly_changes[measure][year] = year_df['Value'].iloc[0]
        else:
            yearly_changes[measure][year] = float('nan')


# Plotting the yearly changes for each unique measure
plt.figure(figsize=(15, 8)) # Adjust the figure size as needed

for measure, values in yearly_changes.items():
    years = list(values.keys())
    changes = list(values.values())
    plt.plot(years, changes, label=measure)
with plt.style.context('ggplot'):
  plt.xlabel('Year')
  plt.ylabel('Value')
  plt.title('Yearly Mean Sea Level Change in Important Water Bodies (1992-2022)')
  plt.legend()
  plt.grid(True)
  plt.show()
```




```
import matplotlib.pyplot as plt

bay_of_bengal_data = yearly_changes.get('Bay Bengal', {})
arabian_sea_data = yearly_changes.get('Arabian Sea', {})
indian_ocean_data = yearly_changes.get('Indian Ocean', {})

plt.figure(figsize=(12, 6))
```

```
plt.plot(list(bay_of_bengal_data.keys()), list(bay_of_bengal_data.values()), label='Bay of Bengal')
plt.plot(list(arabian_sea_data.keys()), list(arabian_sea_data.values()), label='Arabian Sea')
plt.plot(list(indian_ocean_data.keys()), list(indian_ocean_data.values()), label='Indian Ocean')


plt.xlabel('Year')
plt.ylabel('Change in Value')
plt.title('Change in Value for Bay of Bengal, Arabian Sea, and Indian Ocean (1992-2022)')
plt.legend()
plt.grid(True)
plt.xlim(1992, 2022)
plt.show()
```




```
print(arabian_sea_data)
```

```
{1992: -18.46, 1993: 32.14, 1994: 3.84, 1995: -3.66, 1996: 3.64, 1997: -49.66, 1998: 4.04, 1999: -29.46, 2000: -29.96, 2001: -23.56,
```

```
print(indian_ocean_data)
```

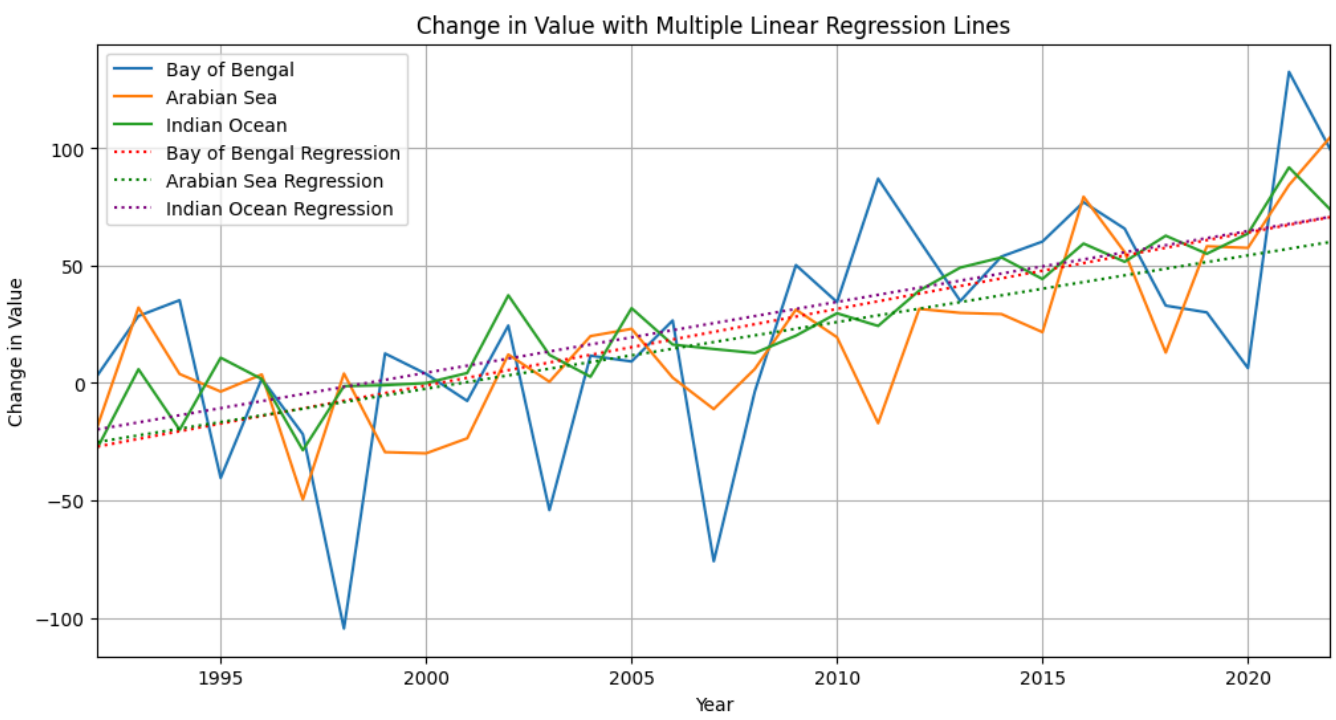
```
{1992: -27.63, 1993: 5.87, 1994: -19.93, 1995: 10.77, 1996: 1.67, 1997: -28.63, 1998: -1.33, 1999: -0.93, 2000: -0.03, 2001: 4.27, 2
```

```
print(bay_of_bengal_data)
```

```
{1992: 3.27, 1993: 28.57, 1994: 35.27, 1995: -40.43, 1996: 1.87, 1997: -21.83, 1998: -104.63, 1999: 12.57, 2000: 3.87, 2001: -7.63,
```

```
import statsmodels.api as sm

bay_of_bengal_data = yearly_changes.get('Bay Bengal', {})
arabian_sea_data = yearly_changes.get('Arabian Sea', {})
indian_ocean_data = yearly_changes.get('Indian Ocean', {})

# Function to perform linear regression and return the regression line
def get_regression_line(data):
    years = list(data.keys())
    values = list(data.values())

    # Remove NaN values
    valid_indices = [i for i, val in enumerate(values) if not pd.isna(val)]
    years_valid = [years[i] for i in valid_indices]
    values_valid = [values[i] for i in valid_indices]

    X = sm.add_constant(years_valid) # Add a constant for the intercept
    model = sm.OLS(values_valid, X).fit()
    return model.params[0], model.params[1]


# Get regression lines for each dataset
```

```
intercept_bay, slope_bay = get_regression_line(bay_of_bengal_data)
intercept_arabian, slope_arabian = get_regression_line(arabian_sea_data)
intercept_indian, slope_indian = get_regression_line(indian_ocean_data)

# Create the plot
plt.figure(figsize=(12, 6))

# Plot each dataset
plt.plot(list(bay_of_bengal_data.keys()), list(bay_of_bengal_data.values()), label='Bay of Bengal')
plt.plot(list(arabian_sea_data.keys()), list(arabian_sea_data.values()), label='Arabian Sea')
plt.plot(list(indian_ocean_data.keys()), list(indian_ocean_data.values()), label='Indian Ocean')


# Plot regression lines
years = range(1992, 2023)
plt.plot(years, [intercept_bay + slope_bay * year for year in years], color='red', label='Bay of Bengal Regression',linestyle='dotted')
plt.plot(years, [intercept_arabian + slope_arabian * year for year in years], color='green', label='Arabian Sea Regression',linestyle='do
plt.plot(years, [intercept_indian + slope_indian * year for year in years], color='purple', label='Indian Ocean Regression',linestyle='do

# Customize the plot
plt.xlabel('Year')
plt.ylabel('Change in Value')
plt.title('Change in Value with Multiple Linear Regression Lines')
plt.legend()
plt.grid(True)
plt.xlim(1992, 2022)  # Set x-axis limits
plt.show()
```




```
import numpy as np

def regression_summary(data, name):
    years = list(data.keys())
    values = list(data.values())

    # Remove NaN values
    valid_indices = [i for i, val in enumerate(values) if not pd.isna(val)]
    years_valid = np.array([years[i] for i in valid_indices])
    values_valid = np.array([values[i] for i in valid_indices])

    X = sm.add_constant(years_valid)
    model = sm.OLS(values_valid, X).fit()
    print(f"Summary for {name}:")
    print(model.summary())

    predictions = model.predict(X)
    mse = np.mean((values_valid - predictions)**2)
    rmse = np.sqrt(mse)
    mae = np.mean(np.abs(values_valid - predictions))
    print(f"Mean Squared Error (MSE): {mse}")
    print(f"Root Mean Squared Error (RMSE): {rmse}")
    print(f"Mean Absolute Error (MAE): {mae}")
    print("-" * 20)
```

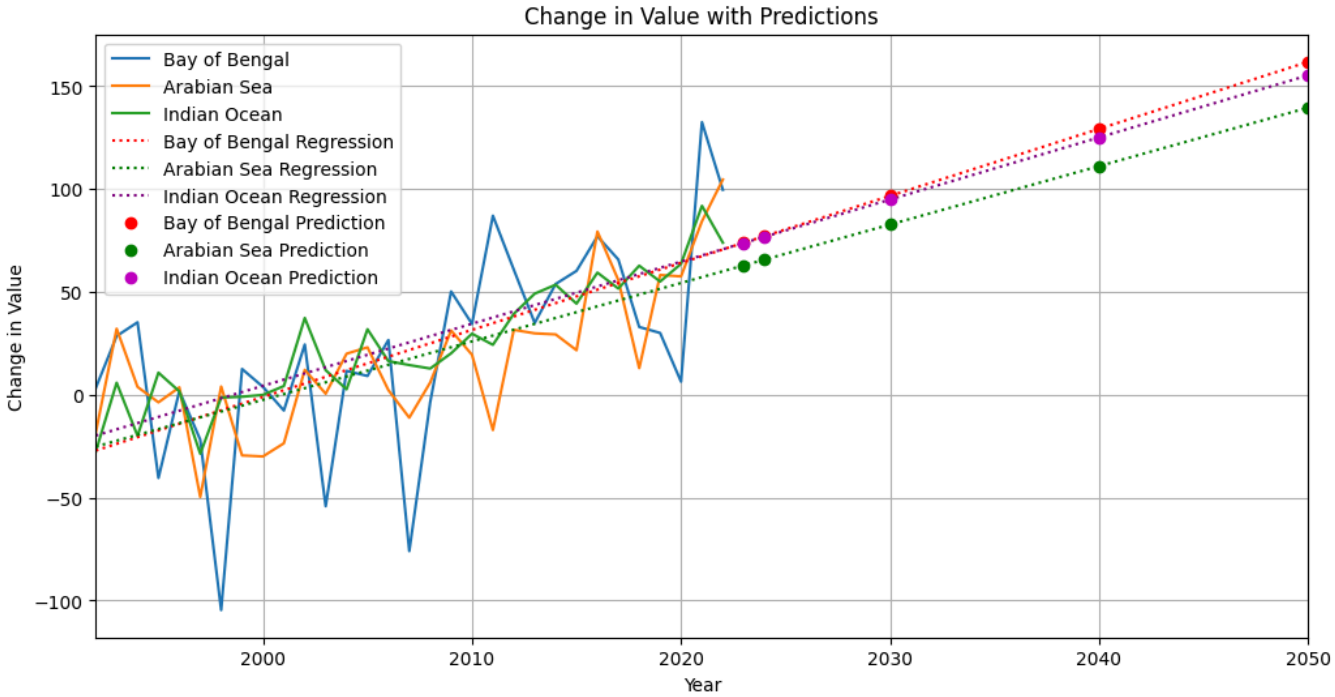
```
regression_summary(bay_of_bengal_data, "Bay of Bengal")
regression_summary(arabian_sea_data, "Arabian Sea")
regression_summary(indian_ocean_data, "Indian Ocean")
```

```
Date:                Sat, 14 Dec 2024   Prob (F-statistic):           2.75e-06
Time:                        03:34:08   Log-Likelihood:                -141.97
No. Observations:                  31   AIC:                             287.9
Df Residuals:                      29   BIC:                             290.8
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -5684.0382    982.836     -5.783      0.000   -7694.163   -3673.913
x1             2.8408      0.490      5.801      0.000       1.839       3.842
==============================================================================
Omnibus:                        0.162   Durbin-Watson:                   1.511
Prob(Omnibus):                  0.922   Jarque-Bera (JB):                0.228
Skew:                           0.150   Prob(JB):                        0.892
Kurtosis:                       2.706   Cond. No.                     4.50e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.5e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
Mean Squared Error (MSE): 556.3479337135805
Root Mean Squared Error (RMSE): 23.587028929341237
Mean Absolute Error (MAE): 18.99426404786677
--------------------
Summary for Indian Ocean:
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.837
Model:                            OLS   Adj. R-squared:                  0.831
Method:                 Least Squares   F-statistic:                     149.0
Date:                Sat, 14 Dec 2024   Prob (F-statistic):           5.97e-13
Time:                        03:34:08   Log-Likelihood:                -120.80
No. Observations:                  31   AIC:                             245.6
Df Residuals:                      29   BIC:                             248.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -6034.4112    496.438    -12.155      0.000   -7049.741   -5019.081
x1             3.0194      0.247     12.207      0.000       2.514       3.525
==============================================================================
Omnibus:                        2.611   Durbin-Watson:                   2.123
Prob(Omnibus):                  0.271   Jarque-Bera (JB):                1.884
Skew:                           0.604   Prob(JB):                        0.390
Kurtosis:                       3.006   Cond. No.                     4.50e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.5e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
Mean Squared Error (MSE): 141.9432567104577
Root Mean Squared Error (RMSE): 11.913994154373993
Mean Absolute Error (MAE): 9.27123803329857
--------------------
```

```
# Predict values for future years
future_years = [2023, 2024, 2030, 2040, 2050]

predicted_bay_of_bengal = [intercept_bay + slope_bay * year for year in future_years]
predicted_arabian_sea = [intercept_arabian + slope_arabian * year for year in future_years]
predicted_indian_ocean = [intercept_indian + slope_indian * year for year in future_years]

# Print the predictions
print("Predictions for future years:")
for i in range(len(future_years)):
    print(f"Year: {future_years[i]}")
    print(f"  Bay of Bengal: {predicted_bay_of_bengal[i]}")
    print(f"  Arabian Sea: {predicted_arabian_sea[i]}")
    print(f"  Indian Ocean: {predicted_indian_ocean[i]}")

# Plot the predictions along with the existing data and regression lines
plt.figure(figsize=(12, 6))

# Plot each dataset
plt.plot(list(bay_of_bengal_data.keys()), list(bay_of_bengal_data.values()), label='Bay of Bengal')
plt.plot(list(arabian_sea_data.keys()), list(arabian_sea_data.values()), label='Arabian Sea')
plt.plot(list(indian_ocean_data.keys()), list(indian_ocean_data.values()), label='Indian Ocean')

# Plot regression lines
```

```
years = range(1992, 2051) # Extended to include future years
plt.plot(years, [intercept_bay + slope_bay * year for year in years], color='red', label='Bay of Bengal Regression', linestyle='dotted')
plt.plot(years, [intercept_arabian + slope_arabian * year for year in years], color='green', label='Arabian Sea Regression', linestyle='d
plt.plot(years, [intercept_indian + slope_indian * year for year in years], color='purple', label='Indian Ocean Regression', linestyle='d

# Plot future predictions
plt.plot(future_years, predicted_bay_of_bengal, 'ro', label='Bay of Bengal Prediction')
plt.plot(future_years, predicted_arabian_sea, 'go', label='Arabian Sea Prediction')
plt.plot(future_years, predicted_indian_ocean, 'mo', label='Indian Ocean Prediction')


# Customize the plot
plt.xlabel('Year')
plt.ylabel('Change in Value')
plt.title('Change in Value with Predictions')
plt.legend()
plt.grid(True)
plt.xlim(1992, 2050)  # Set x-axis limits
plt.show()
```

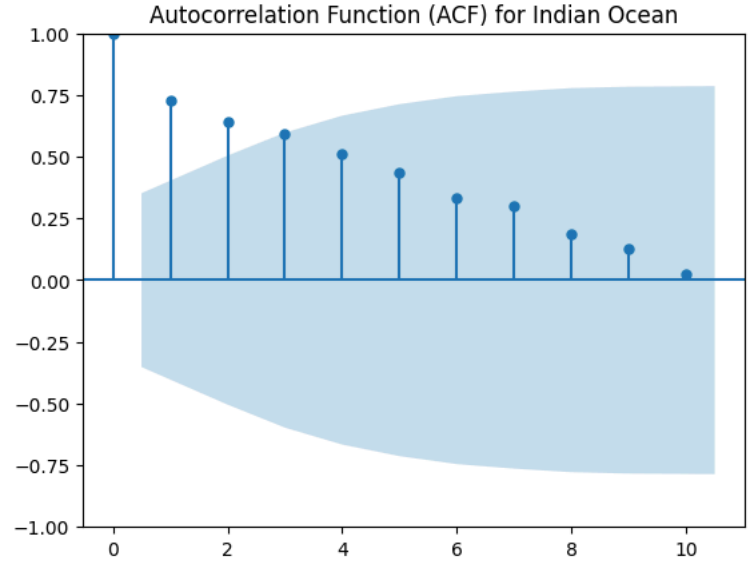
```
Predictions for future years:
Year: 2023
  Bay of Bengal: 73.92012903225714
  Arabian Sea: 62.905096774193225
  Indian Ocean: 73.81703225806359
Year: 2024
  Bay of Bengal: 77.17846370967618
  Arabian Sea: 65.74589919354821
  Indian Ocean: 76.83642338709615
Year: 2030
  Bay of Bengal: 96.72847177419226
  Arabian Sea: 82.79071370967722
  Indian Ocean: 94.95277016128966
Year: 2040
  Bay of Bengal: 129.31181854838542
  Arabian Sea: 111.19873790322526
  Indian Ocean: 125.14668145161158
Year: 2050
  Bay of Bengal: 161.89516532257858
  Arabian Sea: 139.6067620967733
  Indian Ocean: 155.3405927419335
```



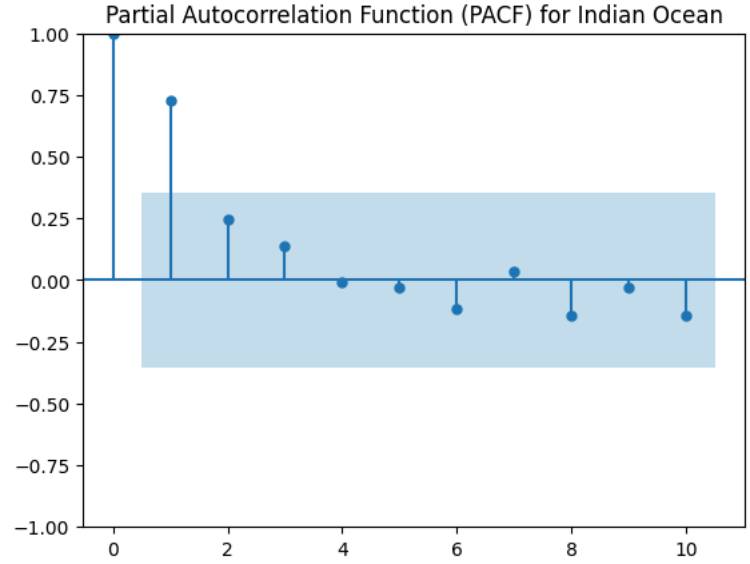

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

indian_ocean_data = yearly_changes.get('Indian Ocean', {})
values = list(indian_ocean_data.values())

# Remove NaN values
values = [x for x in values if not pd.isna(x)]

# Plot ACF
plt.figure(figsize=(10, 5))
plot_acf(values, lags=10, alpha=0.05)  # Adjust lags as needed
```

```
plt.title('Autocorrelation Function (ACF) for Indian Ocean')
plt.show()

# Plot PACF
plt.figure(figsize=(10, 5))
plot_pacf(values, lags=10, alpha=0.05, method='ywm') # Adjust lags as needed
plt.title('Partial Autocorrelation Function (PACF) for Indian Ocean')
plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
```

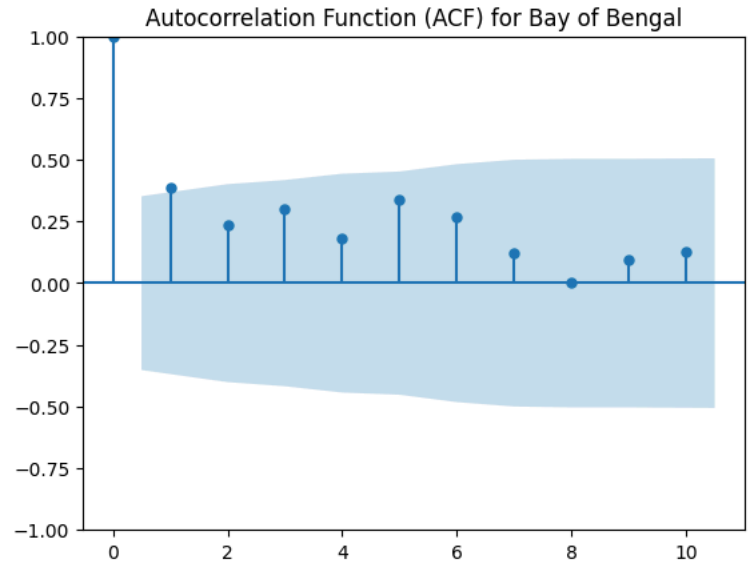



```
<Figure size 1000x500 with 0 Axes>
```

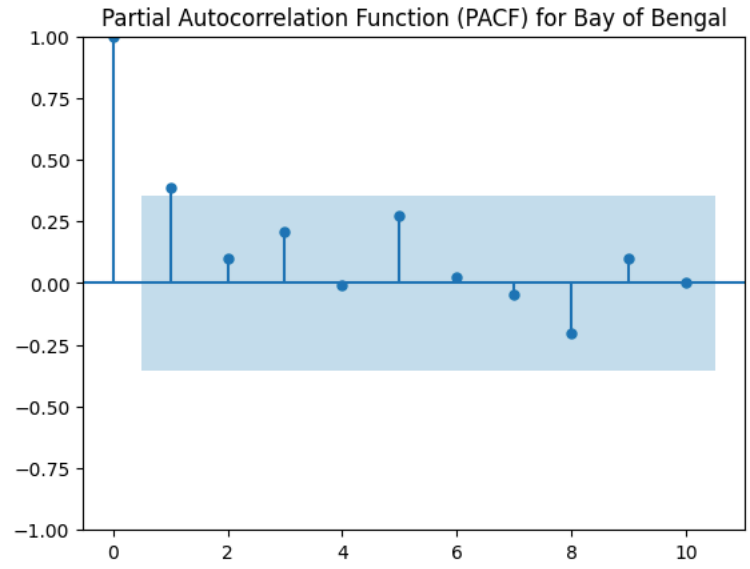



```
bay_of_bengal_data = yearly_changes.get('Bay Bengal', {})
values = list(bay_of_bengal_data.values())

# Remove NaN values
values = [x for x in values if not pd.isna(x)]

# Plot ACF
plt.figure(figsize=(10, 5))
plot_acf(values, lags=10, alpha=0.05)  # Adjust lags as needed
plt.title('Autocorrelation Function (ACF) for Bay of Bengal')
plt.show()

# Plot PACF
plt.figure(figsize=(10, 5))
plot_pacf(values, lags=10, alpha=0.05, method='ywm') # Adjust lags as needed
plt.title('Partial Autocorrelation Function (PACF) for Bay of Bengal')
plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
```

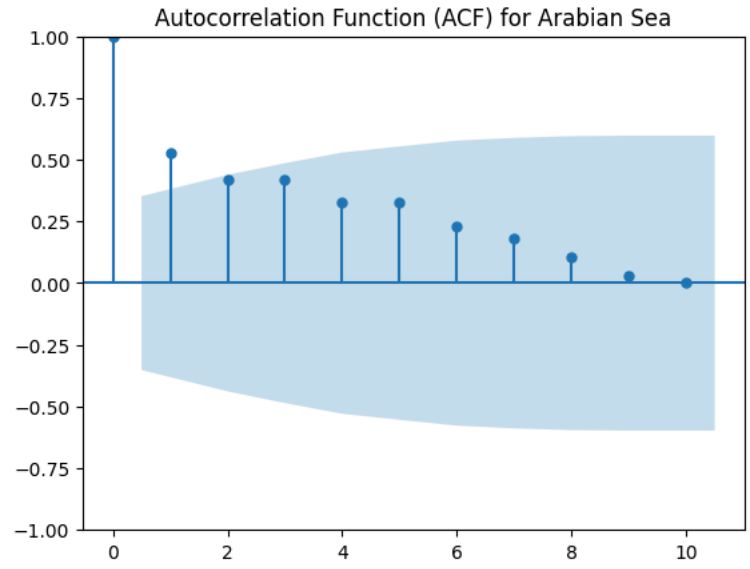



```
<Figure size 1000x500 with 0 Axes>
```

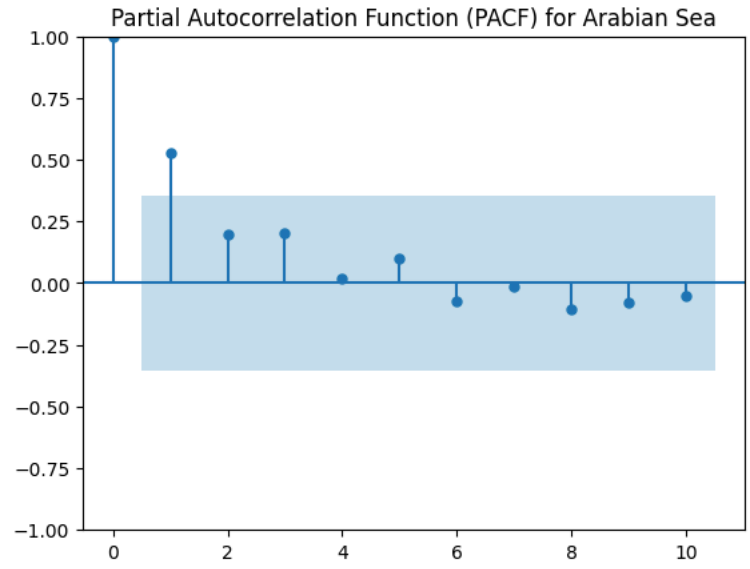



```
arabian_sea_data = yearly_changes.get('Arabian Sea', {})
values = list(arabian_sea_data.values())

# Remove NaN values
values = [x for x in values if not pd.isna(x)]

# Plot ACF
plt.figure(figsize=(10, 5))
plot_acf(values, lags=10, alpha=0.05)  # Adjust lags as needed
plt.title('Autocorrelation Function (ACF) for Arabian Sea')
plt.show()

# Plot PACF
plt.figure(figsize=(10, 5))
plot_pacf(values, lags=10, alpha=0.05, method='ywm') # Adjust lags as needed
plt.title('Partial Autocorrelation Function (PACF) for Arabian Sea')
plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
```




```
<Figure size 1000x500 with 0 Axes>
```




```
from statsmodels.tsa.stattools import adfuller

result = adfuller(values)

# The p-value is in the result[1] position.
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])

# Determine the optimal d value based on the p-value
# We don't need to look for p and q values from the plots here because we are calculating the p-value for the Augmented Dickey Fuller tes

if result[1] <= 0.05:
    print("The time series is likely stationary (d=0).")
    d = 0
else:
    print("The time series is likely non-stationary.")
    print("Difference the series once and repeat ADF test. You might need to difference it multiple times")
    # Difference the series once.
    values_diff = np.diff(values)

    # Perform ADF test on differenced data
    result_diff = adfuller(values_diff)
    print('ADF Statistic (Differenced): %f' % result_diff[0])
    print('p-value (Differenced): %f' % result_diff[1])

    if result_diff[1] <= 0.05:
        print("The differenced time series is likely stationary (d=1).")
        d = 1
    else:
        print("The differenced time series is likely non-stationary.")
        print("Try differencing again (d=2)")
        d = 2 # Or try further differencing until the series becomes stationary
```

```
print(f"The appropriate value for d is: {d}")
```

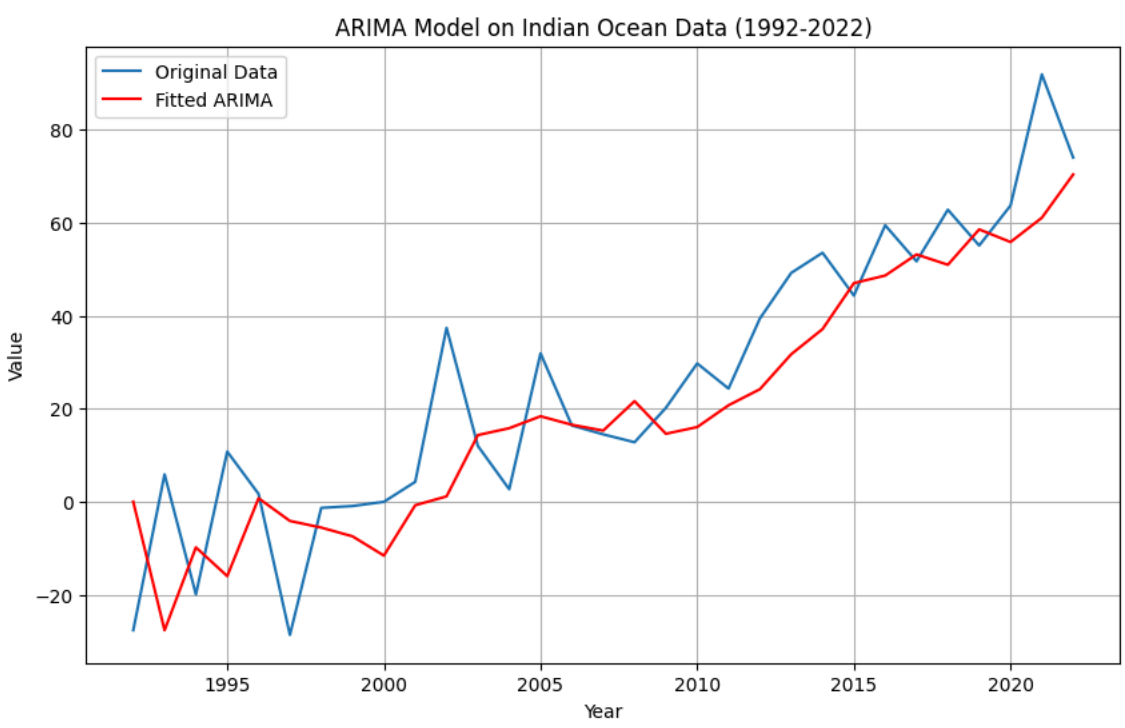
```
ADF Statistic: 0.250121
p-value: 0.974937
The time series is likely non-stationary.
Difference the series once and repeat ADF test. You might need to difference it multiple times
ADF Statistic (Differenced): -5.978864
p-value (Differenced): 0.000000
The differenced time series is likely stationary (d=1).
The appropriate value for d is: 1
```

```
!pip install pmdarima
# Import the necessary library
from pmdarima import auto_arima

indian_ocean_data = yearly_changes.get('Indian Ocean', {})
values = list(indian_ocean_data.values())
values = [x for x in values if not pd.isna(x)]

# Use auto_arima to find the best (p, d, q) order
stepwise_model = auto_arima(values, start_p=0, start_q=0,
                            max_p=5, max_q=5, m=12,
                            start_P=0, seasonal=False,
                            d=None, D=1, trace=True,
                            error_action='ignore',
                            suppress_warnings=True,
                            stepwise=True)

print(stepwise_model.summary())
```

```
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.13.1)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.2.3)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (75.1.0)
Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (24.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2024.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->pmdarima
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (1.0.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas>=0.19->pm
Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (2.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 24.8 MB/s eta 0:00:00
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.4
/usr/local/lib/python3.10/dist-packages/pmdarima/arima/_validation.py:62: UserWarning: m (12) set for non-seasonal fit. Setting t
  warnings.warn("m (%i) set for non-seasonal fit. Setting to 0" % m)
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=261.361, Time=0.06 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=253.678, Time=0.03 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.12 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=260.446, Time=0.01 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=247.449, Time=0.07 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=249.310, Time=0.10 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=249.224, Time=0.09 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.15 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.26 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=252.884, Time=0.03 sec

Best model:  ARIMA(2,1,0)(0,0,0)[0] intercept
Total fit time: 0.962 seconds
                               SARIMAX Results
================================================================================
Dep. Variable:                      y   No. Observations:                   31
Model:               SARIMAX(2, 1, 0)   Log Likelihood                -119.724
Date:                Sat, 14 Dec 2024   AIC                            247.449
Time:                        03:35:43   BIC                            253.053
Sample:                             0   HQIC                           249.242
                                  - 31
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      7.4877      3.107      2.410      0.016       1.397      13.578
ar.L1         -0.7799      0.246     -3.174      0.002      -1.262      -0.298
ar.L2         -0.5269      0.204     -2.589      0.010      -0.926      -0.128
sigma2       166.0032     48.897      3.395      0.001      70.167     261.840
===================================================================================
Ljung-Box (L1) (Q):                   0.04   Jarque-Bera (JB):                 0.29
```

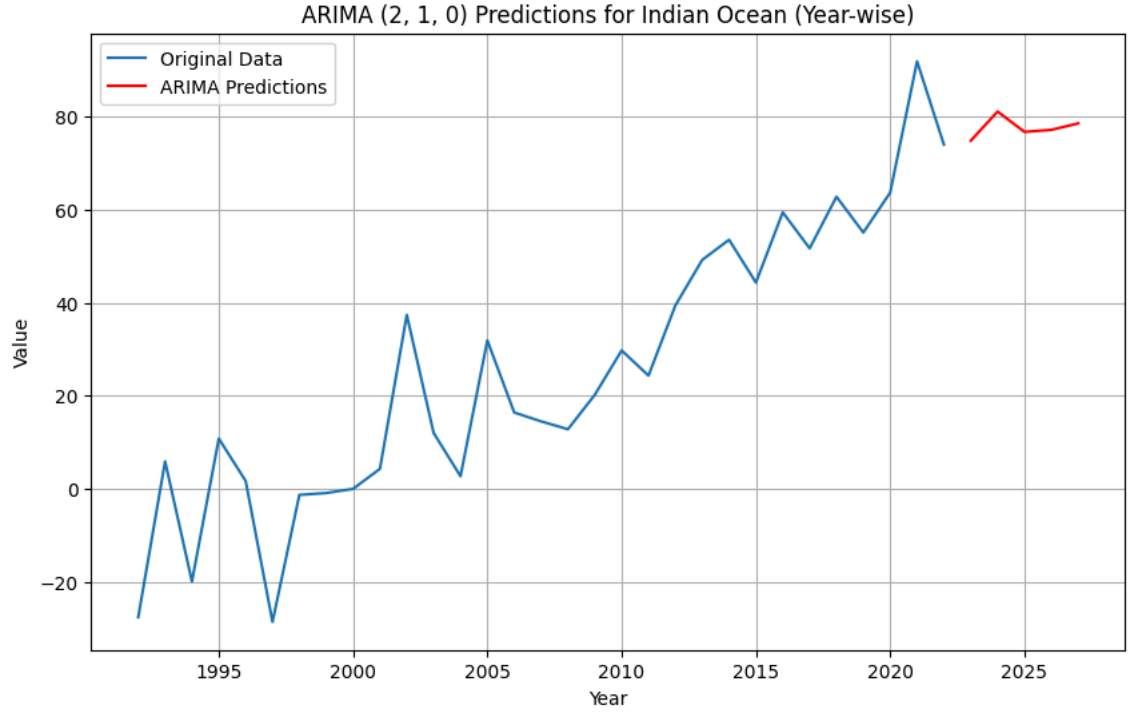
[1] covariance matrix calculated using the outer product of gradients (complex-step).

```
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA

# Assuming indian_ocean_data is defined from the previous code
indian_ocean_data = yearly_changes.get('Indian Ocean', {})
values = list(indian_ocean_data.values())
values = [x for x in values if not pd.isna(x)]
years = list(indian_ocean_data.keys())

# Fit the ARIMA model (example order (2, 1, 0))
model = ARIMA(values, order=(2, 1, 0))
model_fit = model.fit()

# Plot the original data
plt.figure(figsize=(10, 6))
plt.plot(years, values, label='Original Data')

# Plot the fitted ARIMA model
plt.plot(years, model_fit.fittedvalues, label='Fitted ARIMA', color='red')


plt.xlabel('Year')
plt.ylabel('Value')
plt.title('ARIMA Model on Indian Ocean Data (1992-2022)')
plt.legend()
plt.grid(True)
plt.show()
```




```
model = ARIMA(values, order=(2, 1, 0))
model_fit = model.fit()  # This line is crucial - it fits the model and assigns it to model_fit


predictions = model_fit.predict(start=len(values), end=len(values) + 4)

# Plotting the original data and predictions year-wise
plt.figure(figsize=(10, 6))
plt.plot(range(1992, 2023), values, label='Original Data')  # Plot original data with years
future_years = range(2023, 2028)
plt.plot(future_years, predictions, label='ARIMA Predictions', color='red')  # Plot predictions

plt.xlabel('Year')
plt.ylabel('Value')
plt.title('ARIMA (2, 1, 0) Predictions for Indian Ocean (Year-wise)')
plt.legend()
plt.grid(True)
plt.show()
# Print the predictions year-wise (assuming the data starts from 1992)
for i, prediction in enumerate(predictions):
    year = 2023 + i  # Since predictions start from 2023
```

```
    print(f"Year: {year}, Predicted Value: {prediction}")
```



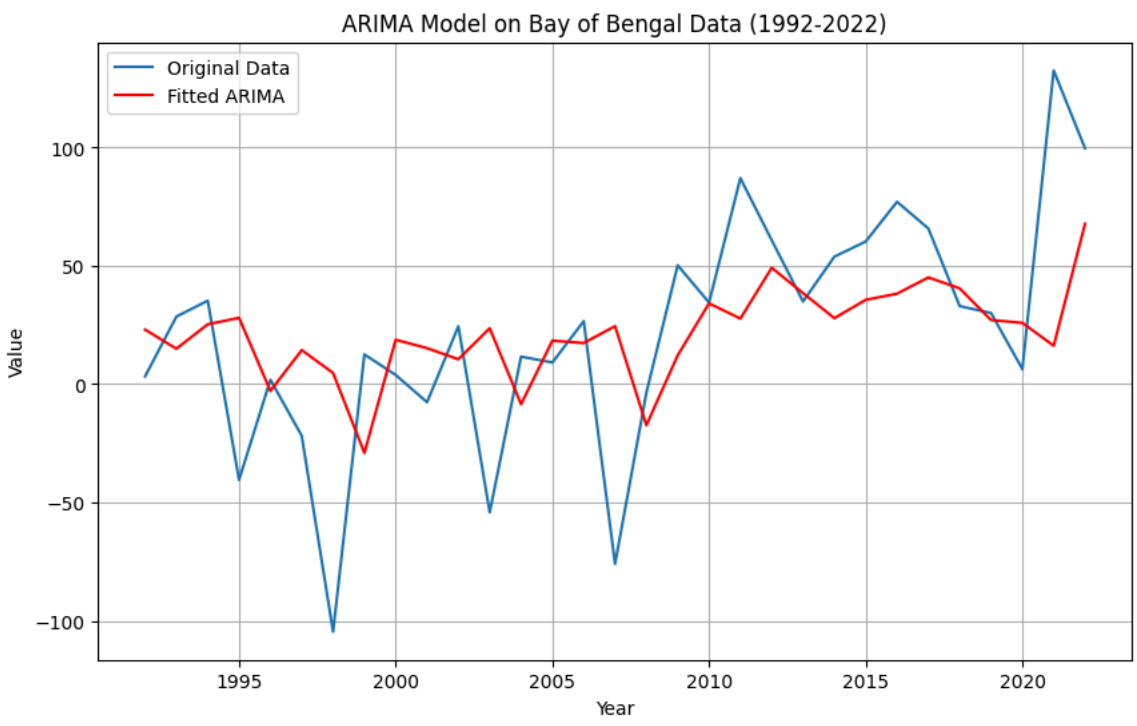

```
Year: 2023, Predicted Value: 74.80899936594646
Year: 2024, Predicted Value: 81.08379345353151
Year: 2025, Predicted Value: 76.70509012136324
Year: 2026, Predicted Value: 77.14779710312341
Year: 2027, Predicted Value: 78.52911964901244
```

```
bay_of_bengal_data = yearly_changes.get('Bay Bengal', {})
values = list(bay_of_bengal_data.values())
values = [x for x in values if not pd.isna(x)]

# Use auto_arima to find the best (p, d, q) order
stepwise_model = auto_arima(values, start_p=0, start_q=0,
                            max_p=5, max_q=5, m=12,
                            start_P=0, seasonal=True,
                            d=None, D=1, trace=True,
                            error_action='ignore',
                            suppress_warnings=True,
                            stepwise=True)

print(stepwise_model.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(0,0,0)(0,1,1)[12] intercept   : AIC=206.176, Time=0.41 sec
 ARIMA(0,0,0)(0,1,0)[12] intercept   : AIC=204.178, Time=0.05 sec
 ARIMA(1,0,0)(1,1,0)[12] intercept   : AIC=206.081, Time=1.37 sec
 ARIMA(0,0,1)(0,1,1)[12] intercept   : AIC=206.679, Time=4.49 sec
 ARIMA(0,0,0)(0,1,0)[12]             : AIC=214.676, Time=0.09 sec
 ARIMA(0,0,0)(1,1,0)[12] intercept   : AIC=206.176, Time=0.86 sec
 ARIMA(0,0,0)(1,1,1)[12] intercept   : AIC=208.176, Time=0.36 sec
 ARIMA(1,0,0)(0,1,0)[12] intercept   : AIC=204.081, Time=0.18 sec
 ARIMA(1,0,0)(0,1,1)[12] intercept   : AIC=206.081, Time=1.11 sec
 ARIMA(1,0,0)(1,1,1)[12] intercept   : AIC=208.081, Time=0.28 sec
 ARIMA(2,0,0)(0,1,0)[12] intercept   : AIC=205.828, Time=0.15 sec
 ARIMA(1,0,1)(0,1,0)[12] intercept   : AIC=205.984, Time=0.15 sec
 ARIMA(0,0,1)(0,1,0)[12] intercept   : AIC=204.681, Time=0.11 sec
 ARIMA(2,0,1)(0,1,0)[12] intercept   : AIC=206.865, Time=0.21 sec
 ARIMA(1,0,0)(0,1,0)[12]             : AIC=206.720, Time=0.04 sec

Best model:  ARIMA(1,0,0)(0,1,0)[12] intercept
Total fit time: 9.886 seconds
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                 y   No. Observations:                   31
Model:             SARIMAX(1, 0, 0)x(0, 1, 0, 12)   Log Likelihood                -99.040
Date:                            Sat, 14 Dec 2024   AIC                            204.081
Time:                                    03:36:12   BIC                            206.914
Sample:                                         0   HQIC                           204.560
                                              - 31
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     30.4637     13.975      2.180      0.029       3.073      57.854
ar.L1          0.3213      0.277      1.158      0.247      -0.223       0.865
```

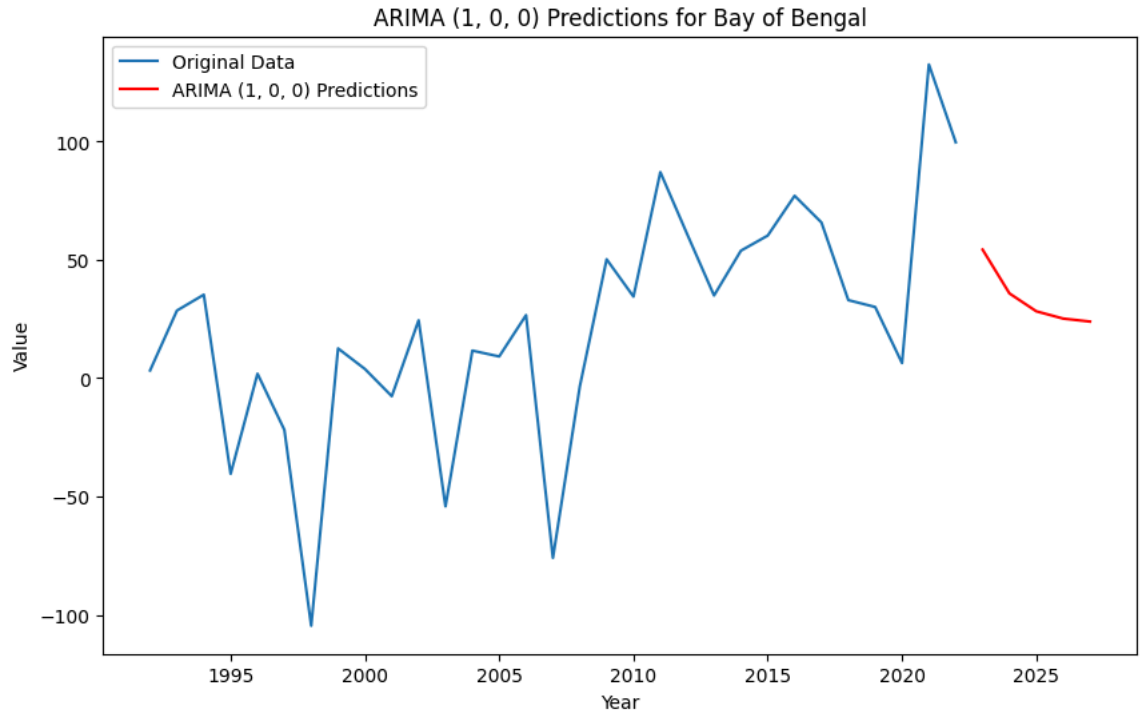
```
sigma2      1962.3022    948.219      2.069      0.039     103.827    3820.777
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):                 1.33
Prob(Q):                              0.90   Prob(JB):                         0.51
Heteroskedasticity (H):               1.10   Skew:                             0.50
Prob(H) (two-sided):                  0.91   Kurtosis:                         2.17
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

```
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA

# Assuming bay_of_bengal_data is defined as in your previous code
bay_of_bengal_data = yearly_changes.get('Bay Bengal', {})
values = list(bay_of_bengal_data.values())
values = [x for x in values if not pd.isna(x)]
years = list(bay_of_bengal_data.keys())
years = years[:len(values)]

# Fit the ARIMA model (example order (1, 0, 0))
model = ARIMA(values, order=(1, 0, 0))
model_fit = model.fit()

# Plot the original data and the fitted values
plt.figure(figsize=(10, 6))
plt.plot(years, values, label='Original Data')
plt.plot(years, model_fit.fittedvalues, label='Fitted ARIMA', color='red')

plt.xlabel('Year')
plt.ylabel('Value')
plt.title('ARIMA Model on Bay of Bengal Data (1992-2022)')
plt.legend()
plt.grid(True)
plt.show()
```




```
bay_of_bengal_data = yearly_changes.get('Bay Bengal', {})
values = list(bay_of_bengal_data.values())
values = [x for x in values if not pd.isna(x)]
years = list(bay_of_bengal_data.keys())
years = years[:len(values)]

# Fit the ARIMA(1, 0, 0) model
model = ARIMA(values, order=(1, 0, 0))  # (p, d, q) order
model_fit = model.fit()

# Make predictions for the next 5 years
predictions = model_fit.predict(start=len(values), end=len(values) + 4)

# Create a list of years for the predictions
future_years = list(range(years[-1] + 1, years[-1] + 6))
```

```
# Plotting
plt.figure(figsize=(10, 6))
plt.plot(years, values, label='Original Data')
plt.plot(future_years, predictions, label='ARIMA (1, 0, 0) Predictions', color='red')

plt.xlabel('Year')
plt.ylabel('Value')
plt.title('ARIMA (1, 0, 0) Predictions for Bay of Bengal')
plt.legend()
plt.show()

# Print the predictions with corresponding years
for year, prediction in zip(future_years, predictions):
    print(f"Predicted value for {year}: {prediction}")
```



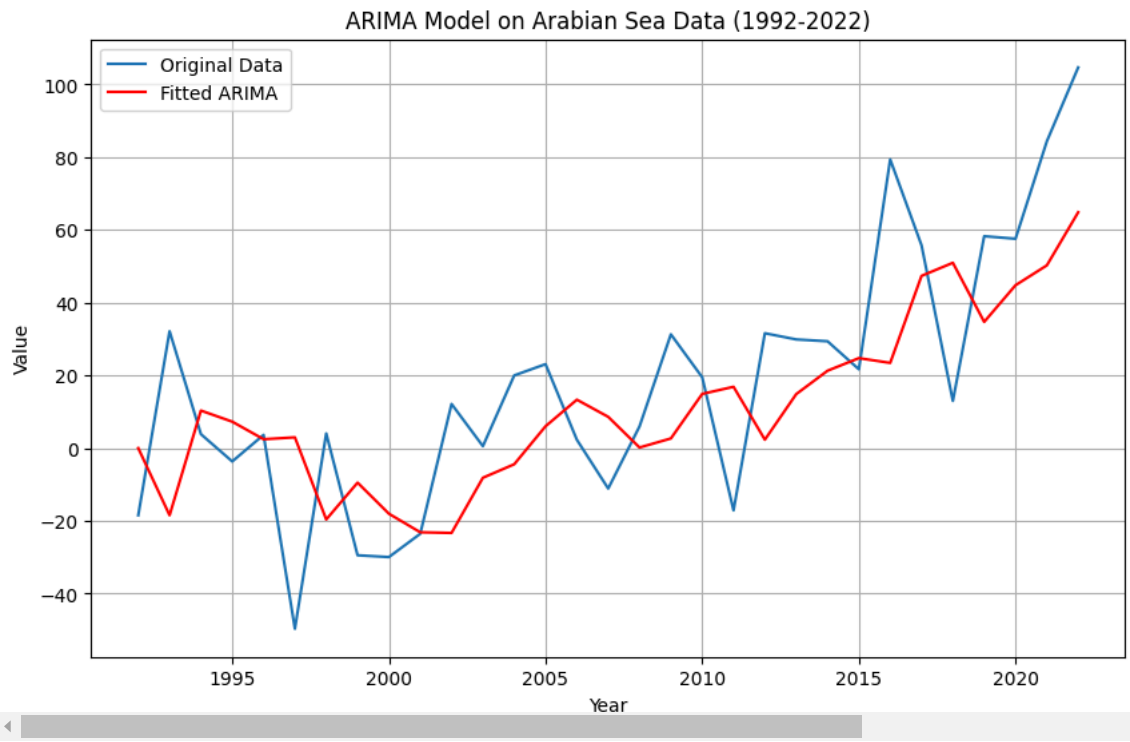

```
Predicted value for 2023: 54.3249113938531
Predicted value for 2024: 35.814727471378234
Predicted value for 2025: 28.2604062392197
Predicted value for 2026: 25.177359116746214
Predicted value for 2027: 23.919115090939002
```

```
from statsmodels.tsa.arima.model import ARIMA

# Assuming 'arabian_sea_data' is defined as in your previous code
arabian_sea_data = yearly_changes.get('Arabian Sea', {})
values = list(arabian_sea_data.values())
values = [x for x in values if not pd.isna(x)]

# Use auto_arima to find the best (p, d, q) order
stepwise_model = auto_arima(values, start_p=0, start_q=0,
                            max_p=5, max_q=5, m=12,
                            start_P=0, seasonal=False,
                            d=None, D=1, trace=False,
                            error_action='ignore',
                            suppress_warnings=True,
                            stepwise=True)

print(stepwise_model.summary())
```

```
/usr/local/lib/python3.10/dist-packages/pmdarima/arima/_validation.py:62: UserWarning: m (12) set for non-seasonal fit. Setting to 0
  warnings.warn("m (%i) set for non-seasonal fit. Setting to 0" % m)
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   31
Model:               SARIMAX(0, 1, 1)   Log Likelihood                -138.910
Date:                Sat, 14 Dec 2024   AIC                            283.821
Time:                        03:36:29   BIC                            288.024
Sample:                             0   HQIC                           285.166
                                  - 31
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
```

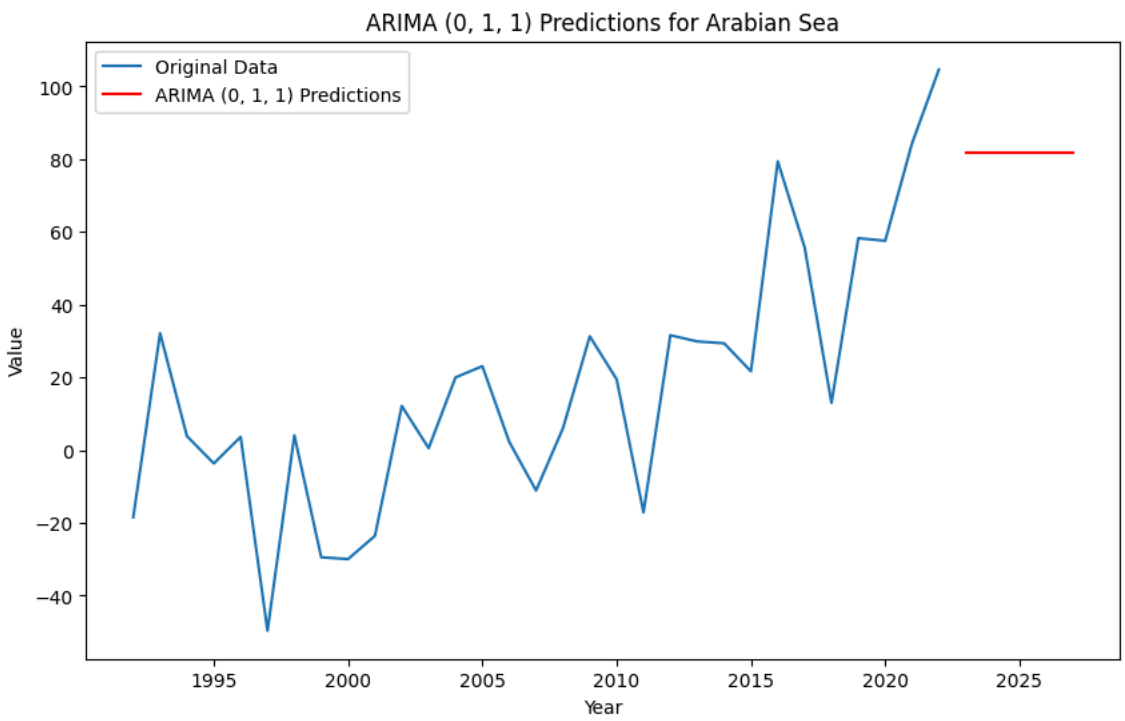
```
intercept      2.9022      1.479      1.962      0.050       0.003       5.801
ma.L1         -0.7161      0.181     -3.951      0.000      -1.071      -0.361
sigma2       601.1656    165.203      3.639      0.000     277.374     924.958
===================================================================================
Ljung-Box (L1) (Q):                   0.05   Jarque-Bera (JB):                 0.43
Prob(Q):                              0.82   Prob(JB):                         0.81
Heteroskedasticity (H):               0.85   Skew:                            -0.28
Prob(H) (two-sided):                  0.80   Kurtosis:                         2.84
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

```
arabian_sea_data = yearly_changes.get('Arabian Sea', {})
values = list(arabian_sea_data.values())
values = [x for x in values if not pd.isna(x)]
years = list(arabian_sea_data.keys())
years = years[:len(values)]

# Fit the ARIMA model (example order (0, 1, 1))
model = ARIMA(values, order=(0, 1, 1))
model_fit = model.fit()

# Plot the original data and the fitted values
plt.figure(figsize=(10, 6))
plt.plot(years, values, label='Original Data')
plt.plot(years, model_fit.fittedvalues, label='Fitted ARIMA', color='red')

plt.xlabel('Year')
plt.ylabel('Value')
plt.title('ARIMA Model on Arabian Sea Data (1992-2022)')
plt.legend()
plt.grid(True)
plt.show()
```




```
arabian_sea_data = yearly_changes.get('Arabian Sea', {})
values = list(arabian_sea_data.values())
values = [x for x in values if not pd.isna(x)]
years = list(arabian_sea_data.keys())
years = years[:len(values)]

# Fit the ARIMA(0, 1, 1) model
model = ARIMA(values, order=(0, 1, 1))
model_fit = model.fit()

# Make predictions for the next 5 years
predictions = model_fit.predict(start=len(values), end=len(values) + 4)

# Create a list of years for the predictions
future_years = list(range(years[-1] + 1, years[-1] + 6))

# Plotting
```

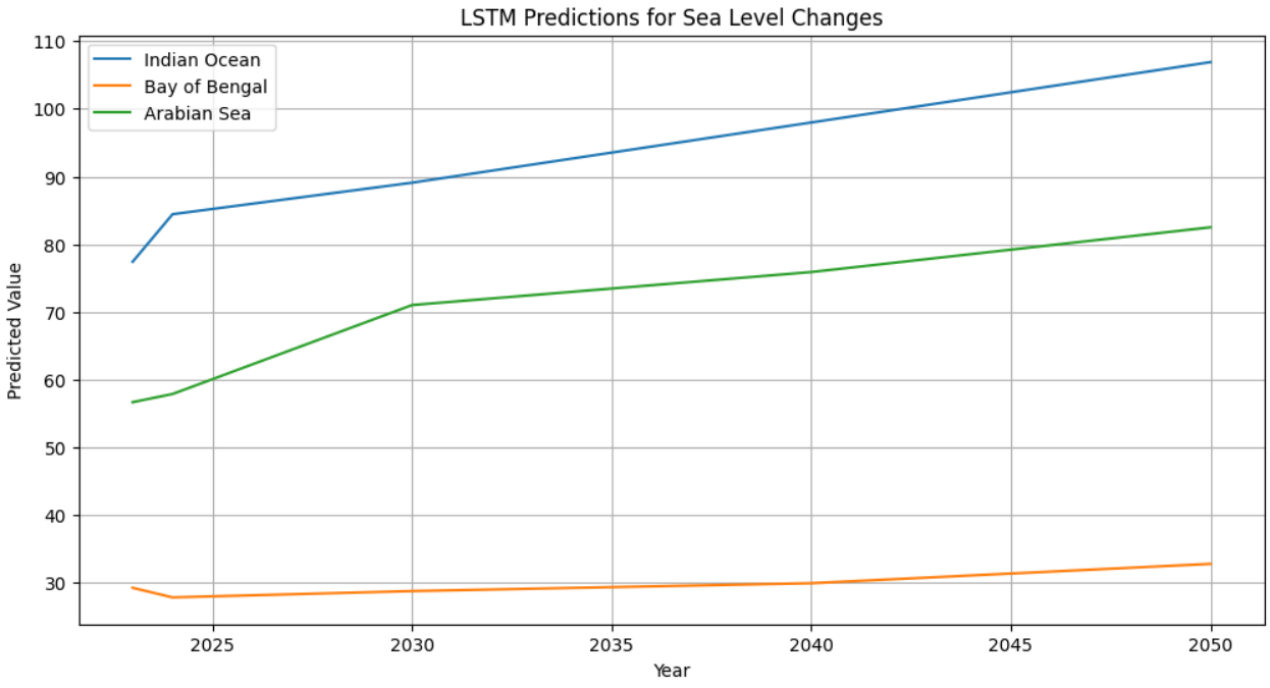
```
# Plotting
plt.figure(figsize=(10, 6))
plt.plot(years, values, label='Original Data')
plt.plot(future_years, predictions, label='ARIMA (0, 1, 1) Predictions', color='red')

plt.xlabel('Year')
plt.ylabel('Value')
plt.title('ARIMA (0, 1, 1) Predictions for Arabian Sea')
plt.legend()
plt.show()

# Print the predictions with corresponding years
print("Predictions for Arabian Sea:")
for year, prediction in zip(future_years, predictions):
    print(f"Predicted value for {year}: {prediction}")
```




```
Predictions for Arabian Sea:
Predicted value for 2023: 81.87949659111074
Predicted value for 2024: 81.87949659111074
Predicted value for 2025: 81.87949659111074
Predicted value for 2026: 81.87949659111074
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import numpy as np


def prepare_data(data):
    values = list(data.values())
    values = [x for x in values if not np.isnan(x)]  # Remove NaN values
    # Normalize the data (important for LSTM)
    values = np.array(values) / np.max(values)

    # Create sequences for LSTM input
    sequence_length = 5  # Adjust as needed
    X, y = [], []
    for i in range(len(values) - sequence_length):
        X.append(values[i:i + sequence_length])
        y.append(values[i + sequence_length])
    X = np.array(X)
    y = np.array(y)
    return X, y

# Prepare data for each region
X_indian, y_indian = prepare_data(indian_ocean_data)
X_bay, y_bay = prepare_data(bay_of_bengal_data)
X_arabian, y_arabian = prepare_data(arabian_sea_data)


# Example LSTM model
def create_lstm_model():
    model = Sequential()
    model.add(LSTM(50, activation='relu', input_shape=(X_indian.shape[1], 1)))  # Input shape adjusted
```

```
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    return model

# Train the model for each region (example)
model_indian = create_lstm_model()
model_indian.fit(X_indian, y_indian, epochs=50, verbose=0)  # Adjust epochs as needed

model_bay = create_lstm_model()
model_bay.fit(X_bay, y_bay, epochs=50, verbose=0)

model_arabian = create_lstm_model()
model_arabian.fit(X_arabian, y_arabian, epochs=50, verbose=0)


# Function to predict values
def predict_future_values(model, X, years):
  predictions = []
  last_sequence = X[-1]
  for _ in range(len(years)):
      prediction = model.predict(np.array([last_sequence]))[0, 0]
      predictions.append(prediction * np.max(list(indian_ocean_data.values())))  # Denormalize
      last_sequence = np.append(last_sequence[1:], prediction)  # Update the sequence
  return predictions


# Predict values for future years
future_years = [2023,2024,2030,2040,2050]
predicted_indian = predict_future_values(model_indian, X_indian, future_years)
print('Indian Ocean data :',predicted_indian)
predicted_bay = predict_future_values(model_bay, X_bay, future_years)
print('Bay of Bengal data :',predicted_bay)
predicted_arabian = predict_future_values(model_arabian, X_arabian, future_years)
print('Arabian Sea data :',predicted_arabian)
# Plot the predictions
plt.figure(figsize=(12, 6))
plt.plot(future_years, predicted_indian, label='Indian Ocean')
plt.plot(future_years, predicted_bay, label='Bay of Bengal')
plt.plot(future_years, predicted_arabian, label='Arabian Sea')

plt.xlabel('Year')
plt.ylabel('Predicted Value')
plt.title('LSTM Predictions for Sea Level Changes')
plt.legend()
plt.grid(True)
plt.show()
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 153ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
Indian Ocean data : [77.41930939435959, 84.43952280461788, 89.09332427620888, 97.9837309718132, 106.89898720502853]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 164ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
Bay of Bengal data : [29.27138907402754, 27.855543135404588, 28.789355684518817, 29.95107430577278, 32.80738573372364]
```

```
print("LSTM Model Summary for Indian Ocean:")
model_indian.summary()

print("\nLSTM Model Summary for Bay of Bengal:")
model_bay.summary()

print("\nLSTM Model Summary for Arabian Sea:")
model_arabian.summary()
```

```
LSTM Model Summary for Indian Ocean:
Model: "sequential_3"
```

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 50)	10,400
dense_3 (Dense)	(None, 1)	51

```
Total params: 31,355 (122.48 KB)
Trainable params: 10,451 (40.82 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 20,904 (81.66 KB)

LSTM Model Summary for Bay of Bengal:
Model: "sequential_4"
```

Layer (type)	Output Shape	Param #
lstm_4 (LSTM)	(None, 50)	10,400
dense_4 (Dense)	(None, 1)	51

```
Total params: 31,355 (122.48 KB)
Trainable params: 10,451 (40.82 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 20,904 (81.66 KB)

LSTM Model Summary for Arabian Sea:
Model: "sequential_5"
```

Layer (type)	Output Shape	Param #
lstm_5 (LSTM)	(None, 50)	10,400
dense_5 (Dense)	(None, 1)	51

```
Total params: 31,355 (122.48 KB)
Trainable params: 10,451 (40.82 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 20,904 (81.66 KB)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd

df = pd.read_csv('/content/drive/My Drive/Project sem1/Datasets/Forest_and_Carbon.csv')

# Now you can work with the dataframe 'df'
print(df.head())
```

```
   ObjectId                       Country ISO2   ISO3  \
0         1            Advanced Economies  NaN  AETMP
1         2            Advanced Economies  NaN  AETMP
2         3            Advanced Economies  NaN  AETMP
3         4            Advanced Economies  NaN  AETMP
4         5  Afghanistan, Islamic Rep. of   AF    AFG

                          Indicator            Unit  \
0           Carbon stocks in forests  Million tonnes
1                      Forest area         1000 HA
2  Index of carbon stocks in forests           Index
3             Index of forest extent           Index
4                      Forest area         1000 HA

                                            Source CTS_Code  CTS_Name  \
0  Food and Agriculture Organization of the Unite...    ECMFC       NaN
1  Food and Agriculture Organization of the Unite...    ECMFF       NaN
2  Food and Agriculture Organization of the Unite...    ECMFS       NaN
3  Food and Agriculture Organization of the Unite...    ECMFE       NaN
4  Food and Agriculture Organization of the Unite...    ECMFF       NaN

                               CTS_Full_Descriptor  ...          F2013  \
0  Environment, Climate Change, Mitigation, Fores...  ...   56891.748000
1  Environment, Climate Change, Mitigation, Fores...  ...  958658.324000
2  Environment, Climate Change, Mitigation, Fores...  ...     108.120860
3  Environment, Climate Change, Mitigation, Fores...  ...     101.355167
4  Environment, Climate Change, Mitigation, Fores...  ...    1208.440000

           F2014          F2015          F2016          F2017          F2018  \
0   57114.846600   57337.946100   57568.174400   57777.849900   57959.762500
1  959818.234200  960978.144400  962013.845500  961845.120000  961963.710000
2     108.544852     108.968845     109.406386     109.804867     110.150586
3     101.477799     101.600432     101.709933     101.692094     101.704632
4    1208.440000    1208.440000    1208.440000    1208.440000    1208.440000

           F2019          F2020          F2021          F2022
0   58035.051300   58102.552100            NaN            NaN
1  962092.030000  962242.860000  962383.873400  962531.435600
2     110.293670     110.421953            NaN            NaN
3     101.718199     101.734145     101.749054     101.764655
4    1208.440000    1208.440000    1208.440000    1208.440000

[5 rows x 41 columns]
```

```
df = df.drop('ISO2', axis=1)
df = df.drop('CTS_Name', axis=1)

print(df.head())
```

```
   ObjectId                       Country   ISO3  \
0         1            Advanced Economies  AETMP
1         2            Advanced Economies  AETMP
2         3            Advanced Economies  AETMP
3         4            Advanced Economies  AETMP
4         5  Afghanistan, Islamic Rep. of    AFG

                          Indicator            Unit  \
0           Carbon stocks in forests  Million tonnes
1                      Forest area         1000 HA
2  Index of carbon stocks in forests           Index
3            Index of forest extent           Index
4                      Forest area         1000 HA

                                          Source CTS_Code  \
0  Food and Agriculture Organization of the Unite...    ECMFC
1  Food and Agriculture Organization of the Unite...    ECMFF
2  Food and Agriculture Organization of the Unite...    ECMFS
3  Food and Agriculture Organization of the Unite...    ECMFE
4  Food and Agriculture Organization of the Unite...    ECMFF

                                CTS_Full_Descriptor        F1992  \
0  Environment, Climate Change, Mitigation, Fores...   52618.6601
1  Environment, Climate Change, Mitigation, Fores...  945840.6089
2  Environment, Climate Change, Mitigation, Fores...     100.0000
3  Environment, Climate Change, Mitigation, Fores...     100.0000
4  Environment, Climate Change, Mitigation, Fores...    1208.4400

           F1993  ...          F2013          F2014          F2015  \
0   53177.699800  ...   56891.748000   57114.846600   57337.946100
1  950628.754500  ...  958658.324000  959818.234200  960978.144400
2     101.062436  ...     108.120860     108.544852     108.968845
3     100.506232  ...     101.355167     101.477799     101.600432
4    1208.440000  ...    1208.440000    1208.440000    1208.440000

           F2016          F2017          F2018          F2019          F2020  \
0   57568.174400   57777.849900   57959.762500   58035.051300   58102.552100
1  962013.845500  961845.120000  961963.710000  962092.030000  962242.860000
2     109.406386     109.804867     110.150586     110.293670     110.421953
3     101.709933     101.692094     101.704632     101.718199     101.734145
4    1208.440000    1208.440000    1208.440000    1208.440000    1208.440000

           F2021          F2022
0            NaN            NaN
1  962383.873400  962531.435600
2            NaN            NaN
3     101.749054     101.764655
4    1208.440000    1208.440000

[5 rows x 39 columns]
```

```
df.dtypes
```

	0
ObjectId	int64
Country	object
ISO3	object
Indicator	object
Unit	object
Source	object
CTS_Code	object
CTS_Full_Descriptor	object
F1992	float64
F1993	float64
F1994	float64
F1995	float64
F1996	float64
F1997	float64
F1998	float64
F1999	float64
F2000	float64
F2001	float64
F2002	float64
F2003	float64
F2004	float64
F2005	float64
F2006	float64
F2007	float64
F2008	float64
F2009	float64
F2010	float64
F2011	float64
F2012	float64
F2013	float64
F2014	float64

```
# Rename columns
new_columns = {}
```

```
for col in df.columns:
  if col.startswith('F') and col[1:].isdigit():
    new_columns[col] = col[1:]
df = df.rename(columns=new_columns)

print(df.head())
df.dtypes
```

```
   ObjectId                       Country   ISO3  \
0         1            Advanced Economies  AETMP
1         2            Advanced Economies  AETMP
2         3            Advanced Economies  AETMP
3         4            Advanced Economies  AETMP
4         5  Afghanistan, Islamic Rep. of    AFG

                          Indicator            Unit  \
0          Carbon stocks in forests  Million tonnes
1                      Forest area         1000 HA
2  Index of carbon stocks in forests           Index
3            Index of forest extent           Index
4                      Forest area         1000 HA

                                            Source CTS_Code  \
0  Food and Agriculture Organization of the Unite...    ECMFC
1  Food and Agriculture Organization of the Unite...    ECMFF
2  Food and Agriculture Organization of the Unite...    ECMFS
3  Food and Agriculture Organization of the Unite...    ECMFE
4  Food and Agriculture Organization of the Unite...    ECMFF

                               CTS_Full_Descriptor         1992  \
0  Environment, Climate Change, Mitigation, Fores...   52618.6601
1  Environment, Climate Change, Mitigation, Fores...  945840.6089
2  Environment, Climate Change, Mitigation, Fores...     100.0000
3  Environment, Climate Change, Mitigation, Fores...     100.0000
4  Environment, Climate Change, Mitigation, Fores...    1208.4400

           1993  ...           2013           2014           2015  \
0   53177.699800  ...   56891.748000   57114.846600   57337.946100
1  950628.754500  ...  958658.324000  959818.234200  960978.144400
2     101.062436  ...     108.120860     108.544852     108.968845
3     100.506232  ...     101.355167     101.477799     101.600432
4    1208.440000  ...    1208.440000    1208.440000    1208.440000

           2016           2017           2018           2019           2020  \
0   57568.174400   57777.849900   57959.762500   58035.051300   58102.552100
1  962013.845500  961845.120000  961963.710000  962092.030000  962242.860000
2     109.406386     109.804867     110.150586     110.293670     110.421953
3     101.709933     101.692094     101.704632     101.718199     101.734145
4    1208.440000    1208.440000    1208.440000    1208.440000    1208.440000

           2021           2022
0           NaN            NaN
1  962383.873400  962531.435600
2           NaN            NaN
3     101.749054     101.764655
4    1208.440000    1208.440000

[5 rows x 39 columns]
```

	0
ObjectId	int64
Country	object
ISO3	object

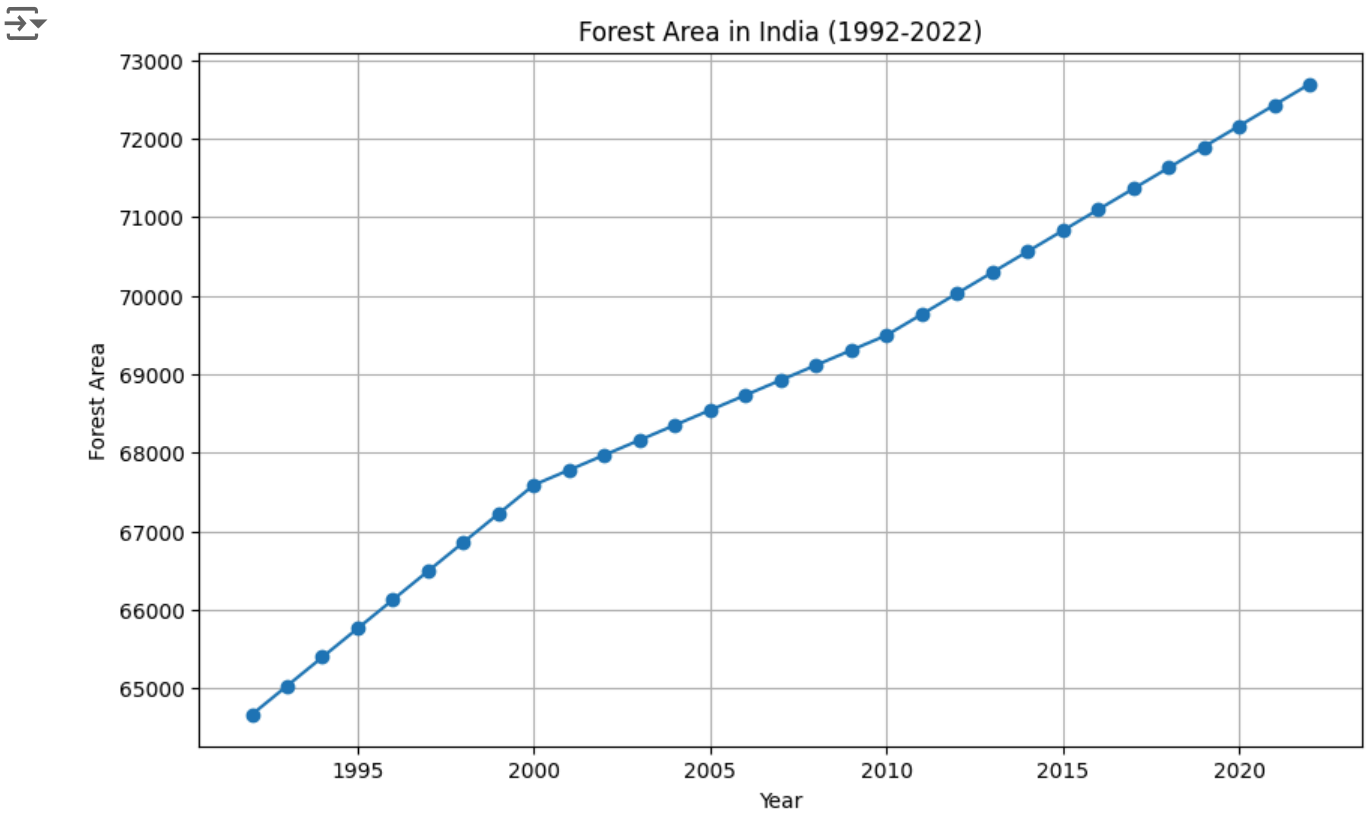
```
# Find missing values where country is India and indicator is forest area
forest_area_india = df[(df['Country'] == 'India') & (df['Indicator'] == 'Forest area')].i
```

```
forest_area_india
```

	0
ObjectId	0
Country	0
ISO3	0
Indicator	0
Unit	0
Source	0
CTS_Code	0
CTS_Full_Descriptor	0
1992	0
1993	0
1994	0
1995	0
1996	0
1997	0
1998	0
1999	0
2000	0
2001	0
2002	0
2003	0
2004	0
2005	0
2006	0
2007	0
2008	0
2009	0
2010	0
2011	0
2012	0
2013	0
2014	0

dtype: int64

```
import matplotlib.pyplot as plt
# Filter data for India and Forest Area
india_forest_data = df[(df['Country'] == 'India') & (df['Indicator'] == 'Forest area')]

# Extract years and forest area values
years = [int(col) for col in india_forest_data.columns if col.isdigit() and 1992 <= int(c
forest_area = [india_forest_data[str(year)].iloc[0] for year in years]

# Create the plot
plt.figure(figsize=(10, 6))
plt.plot(years, forest_area, marker='o')
plt.xlabel('Year')
plt.ylabel('Forest Area')
plt.title('Forest Area in India (1992-2022)')
plt.grid(True)
plt.show()
```

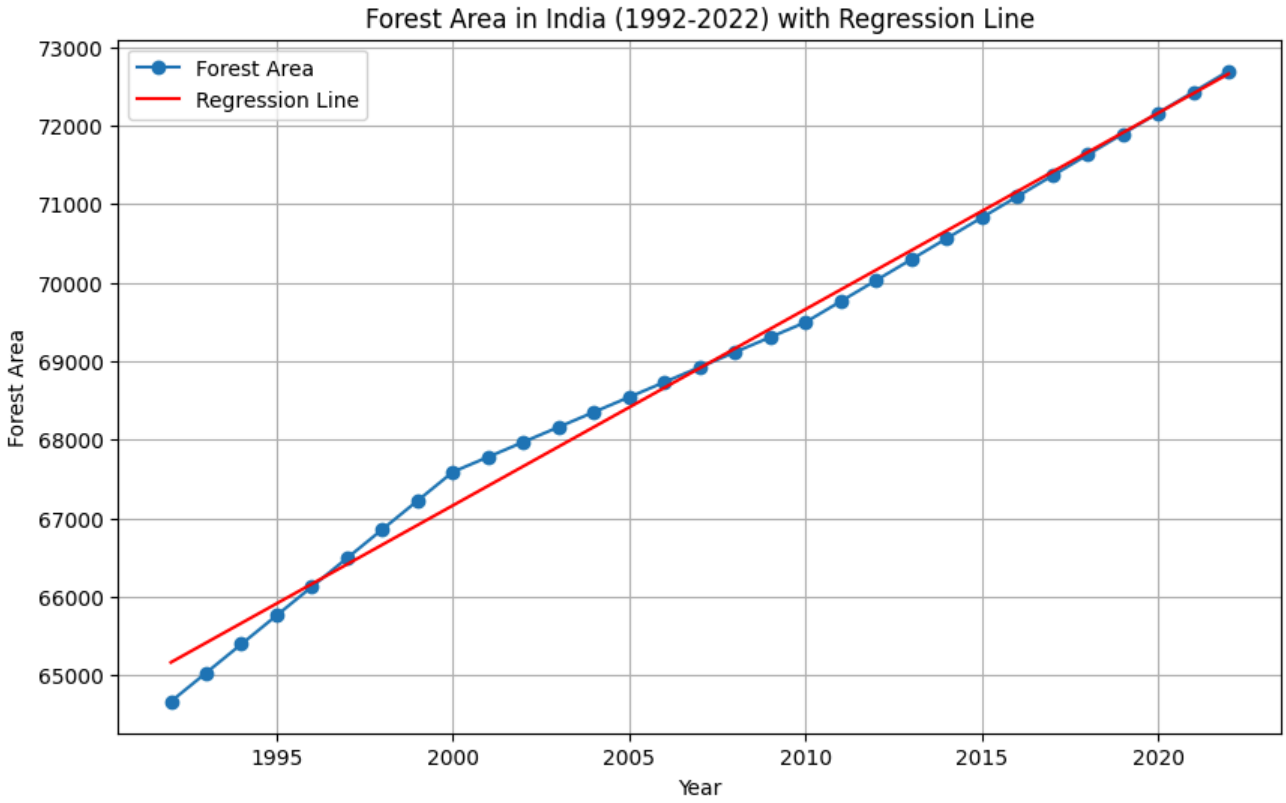



```
print(india_forest_data)
```

```
      ObjectId Country ISO3    Indicator     Unit  \
547        548   India  IND  Forest area  1000 HA

                                                Source CTS_Code  \
547  Food and Agriculture Organization of the Unite...    ECMFF
```

```
                                    CTS_Full_Descriptor     1992     1993  ...  \
547  Environment, Climate Change, Mitigation, Fores...  64668.6  65033.9  ...

        2013     2014     2015     2016     2017     2018     2019     2020  \
547  70295.2  70561.6  70828.0  71094.4  71360.8  71627.2  71893.6  72160.0

        2021     2022
547  72426.4  72692.8

[1 rows x 39 columns]
```

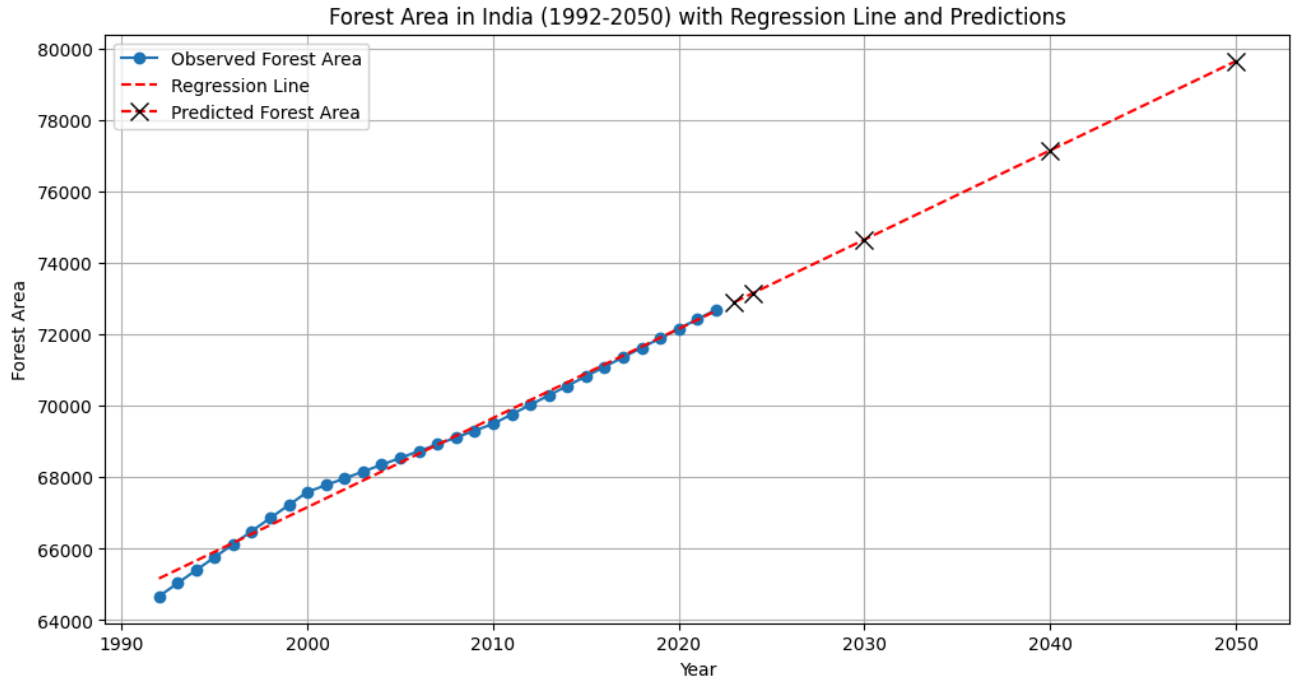
```
import numpy as np
from scipy.stats import linregress

# Calculate linear regression
slope, intercept, r_value, p_value, std_err = linregress(years, forest_area)

# Create the plot
plt.figure(figsize=(10, 6))
plt.plot(years, forest_area, marker='o', label='Forest Area')

# Add regression line
plt.plot(years, intercept + slope * np.array(years), 'r', label='Regression Line')

plt.xlabel('Year')
plt.ylabel('Forest Area')
plt.title('Forest Area in India (1992-2022) with Regression Line')
plt.grid(True)
plt.legend()
plt.show()
```




```
# Print the regression results
print(f"Slope: {slope}")
print(f"Intercept: {intercept}")
print(f"R-value: {r_value*100}%")
print(f"P-value: {p_value}")
print(f"Standard Error: {std_err}")

# R-squared value
r_squared = r_value**2
print(f"R-squared: {r_squared}")


# Summary of the linear regression
print("\nSummary of Linear Regression:")
print(f"The linear regression model shows a relationship between the year and forest area
print(f"The slope ({slope:.2f}) indicates the average annual change in forest area.")
print(f"A positive slope suggests an increasing trend, while a negative slope suggests a
print(f"The intercept ({intercept:.2f}) represents the estimated forest area at year 0 (n
print(f"The R-squared value ({r_squared:.2f}) indicates the proportion of variance in for
print(f"A higher R-squared value indicates a better fit of the model.")
print(f"The p-value ({p_value:.3f}) indicates the statistical significance of the relatio
```

```
Slope: 249.69532258064515
Intercept: -432226.0317741935
R-value: 99.58763709339708%
P-value: 8.753224918248456e-32
Standard Error: 4.223889949234077
R-squared: 0.9917697461846157

Summary of Linear Regression:
The linear regression model shows a relationship between the year and forest area in
The slope (249.70) indicates the average annual change in forest area.
A positive slope suggests an increasing trend, while a negative slope suggests a decr
The intercept (-432226.03) represents the estimated forest area at year 0 (not meanin
The R-squared value (0.99) indicates the proportion of variance in forest area explai
A higher R-squared value indicates a better fit of the model.
The p-value (0.000) indicates the statistical significance of the relationship. A sma
```

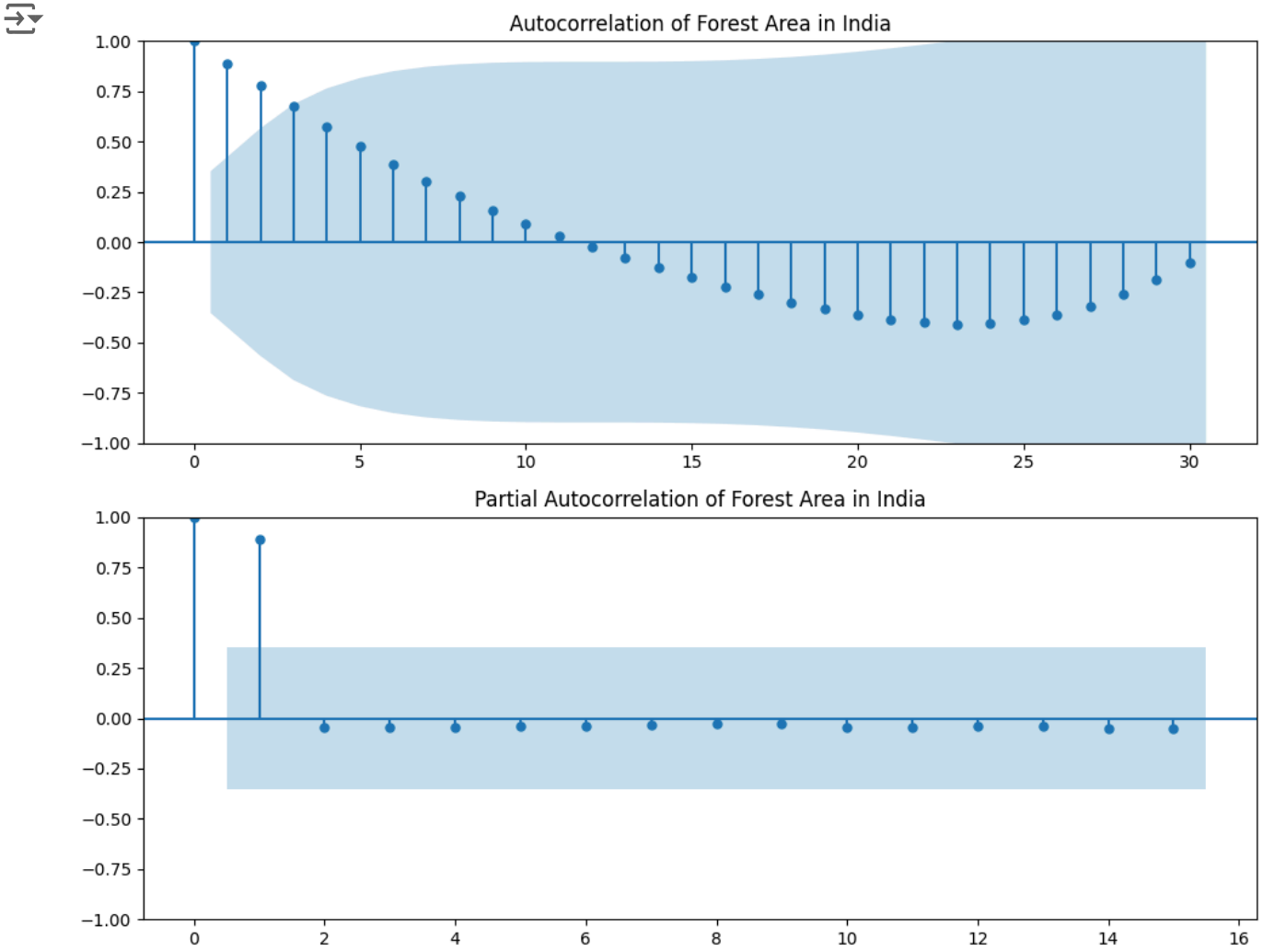
```
# Predict forest area for future years
future_years = [2023, 2024, 2030, 2040, 2050]
predicted_forest_area = intercept + slope * np.array(future_years)

# Extend the plot with predicted values
plt.figure(figsize=(12, 6))
plt.plot(years, forest_area, marker='o', label='Observed Forest Area')
plt.plot(years, intercept + slope * np.array(years), 'r', label='Regression Line',linesty
plt.plot(future_years, predicted_forest_area, marker='x', linestyle='--', color='r', labe


plt.xlabel('Year')
plt.ylabel('Forest Area')
plt.title('Forest Area in India (1992-2050) with Regression Line and Predictions')
plt.grid(True)
plt.legend()
plt.show()

# Print the predicted values
for year, area in zip(future_years, predicted_forest_area):
    print(f"Predicted Forest Area for {year}: {area:.2f}")
```




```
Predicted Forest Area for 2023: 72907.61
Predicted Forest Area for 2024: 73157.30
Predicted Forest Area for 2030: 74655.47
Predicted Forest Area for 2040: 77152.43
Predicted Forest Area for 2050: 79649.38
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Calculate autocorrelation and partial autocorrelation
acf_vals = np.correlate(forest_area, forest_area, mode='full')

# Plotting
fig, axes = plt.subplots(2, 1, figsize=(10, 8))

# Autocorrelation plot using statsmodels
plot_acf(forest_area, lags=len(years)-1, ax=axes[0])
axes[0].set_title('Autocorrelation of Forest Area in India')

# Partial autocorrelation plot using statsmodels
# Limit lags to 50% of the sample size
max_lags = len(years) // 2  # Calculate maximum allowed lags
plot_pacf(forest_area, lags=max_lags, ax=axes[1]) # Changed to calculate maximum lags
```

```
axes[1].set_title('Partial Autocorrelation of Forest Area in India')


plt.tight_layout()
plt.show()
```




```
from statsmodels.tsa.stattools import adfuller

for d in range(3):
    if d > 0:
      differenced_series = np.diff(forest_area, n=d)
```

```
    else:
      differenced_series = forest_area

    result = adfuller(differenced_series)
    print(f"Results for d = {d}")
    print('ADF Statistic: %f' % result[0])
    print('p-value: %f' % result[1])
    print('Critical Values:')
    for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))

    if result[1] <= 0.05:
        print("Reject the null hypothesis. Time series is stationary.")
    else:
        print("Fail to reject the null hypothesis. Time series is not stationary.")
    print("-" * 50)
```

```
Results for d = 0
ADF Statistic: -0.102038
p-value: 0.949235
Critical Values:
        1%: -3.679
        5%: -2.968
        10%: -2.623
Fail to reject the null hypothesis. Time series is not stationary.
--------------------------------------------------
Results for d = 1
ADF Statistic: -5.346733
p-value: 0.000004
Critical Values:
        1%: -3.809
        5%: -3.022
        10%: -2.651
Reject the null hypothesis. Time series is stationary.
--------------------------------------------------
Results for d = 2
ADF Statistic: -5.749507
p-value: 0.000001
Critical Values:
        1%: -3.833
        5%: -3.031
        10%: -2.656
Reject the null hypothesis. Time series is stationary.
--------------------------------------------------
```

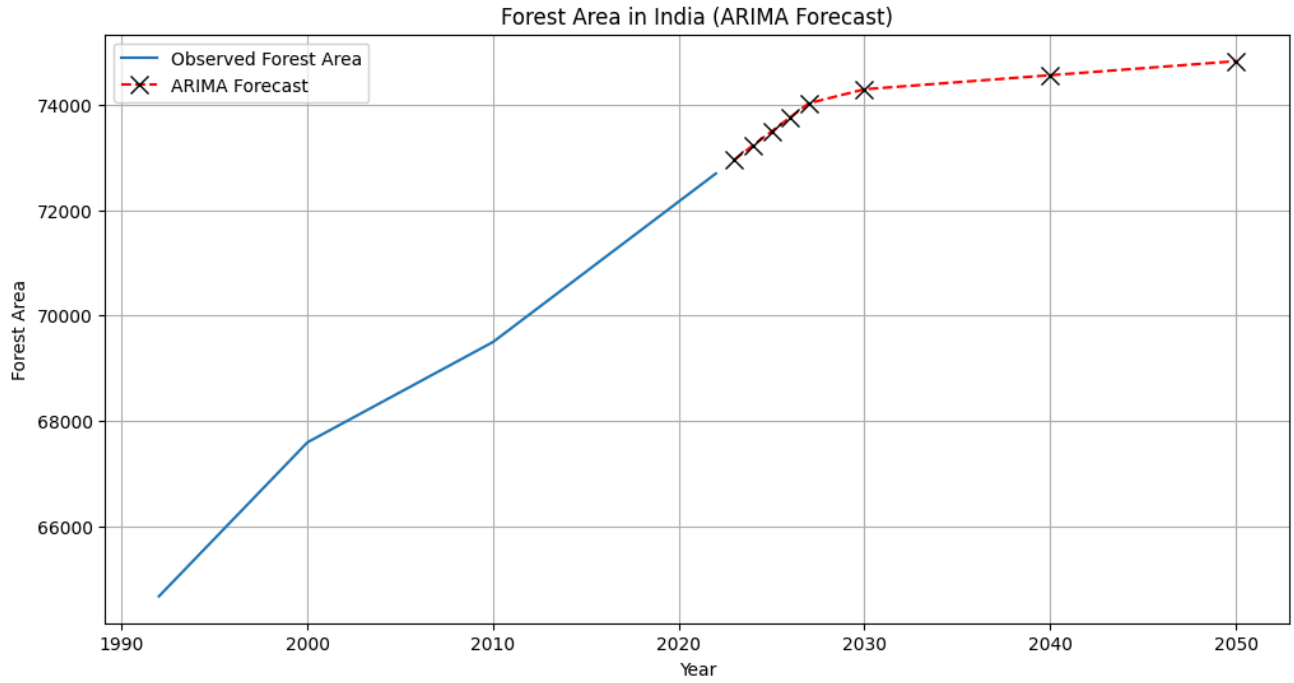
```
!pip install pmdarima
from statsmodels.tsa.arima.model import ARIMA
import pmdarima as pm
# Automatic ARIMA model selection using pmdarima
auto_arima_model = pm.auto_arima(forest_area,
                                 start_p=0, start_q=0,
                                 test='adf',
                                 max_p=5, max_q=5, m=1, # Set appropriate max values
                                 d=None,           # Let auto_arima determine 'd'
                                 seasonal=False,   # No seasonality assumed in this examp
                                 start_P=0,
```

```
                                D=0,
                                trace=True,
                                error_action='ignore',
                                suppress_warnings=True,
                                stepwise=True)

print(auto_arima_model.summary())

# Access the p, d, and q values:
p, d, q = auto_arima_model.order
print(f"Optimal p, d, q values: p={p}, d={d}, q={q}")
```

```
Collecting pmdarima
  Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.m
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/pyt
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.
Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/di
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (f
Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.man
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 21.7 MB/s eta 0:00:00
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.4
Performing stepwise search to minimize aic
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=292.877, Time=0.07 sec
 ARIMA(1,2,0)(0,0,0)[0] intercept   : AIC=294.869, Time=0.05 sec
 ARIMA(0,2,1)(0,0,0)[0] intercept   : AIC=294.869, Time=0.56 sec
 ARIMA(0,2,0)(0,0,0)[0]             : AIC=291.147, Time=0.02 sec
 ARIMA(1,2,1)(0,0,0)[0] intercept   : AIC=296.856, Time=0.99 sec

Best model:  ARIMA(0,2,0)(0,0,0)[0]
Total fit time: 1.720 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   31
Model:               SARIMAX(0, 2, 0)   Log Likelihood                -144.574
Date:                Sat, 14 Dec 2024   AIC                            291.147
Time:                        03:59:23   BIC                            292.514
Sample:                             0   HQIC                           291.575
                                  - 31
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2      1252.2601    103.327     12.119      0.000    1049.743    1454.777
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):               420.07
Prob(Q):                              0.96   Prob(JB):                         0.00
```

```
Heteroskedasticity (H):               0.00   Skew:                            -3.58
Prob(H) (two-sided):                  0.00   Kurtosis:                        20.22
================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
Optimal p, d, q values: p=0, d=2, q=0
```

```
model = ARIMA(forest_area, order=(0, 2, 0))
model_fit = model.fit()

# Forecast future values
future_years = [2023, 2024, 2025, 2026, 2027, 2030, 2040, 2050]
forecast = model_fit.predict(start=len(forest_area), end=len(forest_area) + len(future_ye

# Plotting the forecast
plt.figure(figsize=(12, 6))
plt.plot(years, forest_area, label='Observed Forest Area')
plt.plot(future_years, forecast, label='ARIMA Forecast', marker='x', linestyle='--', colo

plt.xlabel('Year')
plt.ylabel('Forest Area')
plt.title('Forest Area in India (ARIMA Forecast)')
plt.legend()
plt.grid(True)
plt.show()

# Print the forecasted values
for year, area in zip(future_years, forecast):
    print(f"Forecasted Forest Area for {year}: {area:.2f}")
```




```
Forecasted Forest Area for 2023: 72959.20
Forecasted Forest Area for 2024: 73225.60
Forecasted Forest Area for 2025: 73492.00
Forecasted Forest Area for 2026: 73758.40
Forecasted Forest Area for 2027: 74024.80
Forecasted Forest Area for 2030: 74291.20
Forecasted Forest Area for 2040: 74557.60
Forecasted Forest Area for 2050: 74824.00
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense


# Data scaling
scaler = MinMaxScaler()
forest_area_scaled = scaler.fit_transform(np.array(forest_area).reshape(-1, 1))

# Prepare data for LSTM
def create_dataset(dataset, look_back=1):
    X, Y = [], []
    for i in range(len(dataset)-look_back-1):
        a = dataset[i:(i+look_back), 0]
```

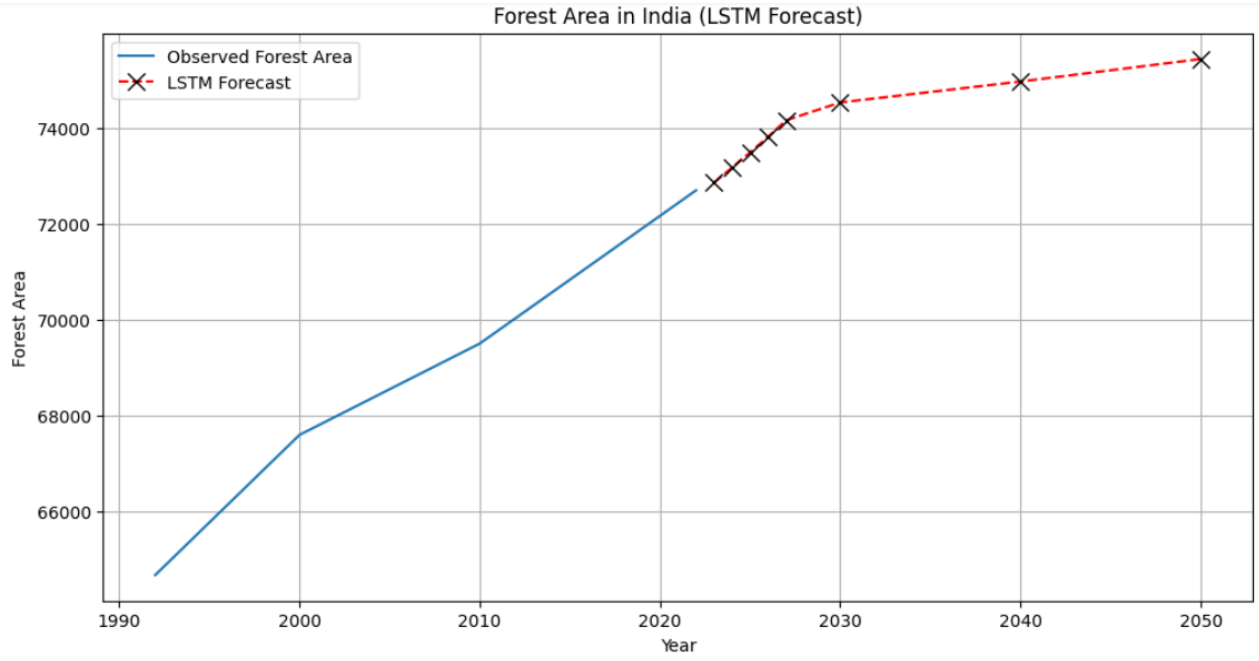
```
        X.append(a)
        Y.append(dataset[i + look_back, 0])
    return np.array(X), np.array(Y)

look_back = 5
X, Y = create_dataset(forest_area_scaled, look_back)
X = np.reshape(X, (X.shape[0], X.shape[1], 1))

# LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X.shape[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(X, Y, epochs=100, batch_size=1, verbose=2)

# Predictions
future_years = [2023, 2024, 2025, 2026, 2027, 2030, 2040, 2050]
inputs = forest_area_scaled[-look_back:]
predictions = []

for i in range(len(future_years)):
    input_reshaped = np.reshape(inputs, (1, look_back, 1))
    prediction = model.predict(input_reshaped, verbose=0)
    predictions.append(prediction[0,0])
    inputs = np.append(inputs[1:], prediction)

# Inverse transform to get actual values
predictions = scaler.inverse_transform(np.array(predictions).reshape(-1, 1))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: UserWarni
  super().__init__(**kwargs)
Epoch 1/100
25/25 - 10s - 397ms/step - loss: 0.1434
Epoch 2/100
25/25 - 0s - 13ms/step - loss: 0.0129
Epoch 3/100
25/25 - 1s - 24ms/step - loss: 0.0026
Epoch 4/100
25/25 - 0s - 9ms/step - loss: 0.0010
Epoch 5/100
25/25 - 0s - 15ms/step - loss: 8.2774e-04
Epoch 6/100
25/25 - 0s - 14ms/step - loss: 8.0157e-04
Epoch 7/100
25/25 - 0s - 10ms/step - loss: 4.6154e-04
Epoch 8/100
25/25 - 0s - 11ms/step - loss: 3.4955e-04
Epoch 9/100
25/25 - 0s - 16ms/step - loss: 3.4022e-04
Epoch 10/100
25/25 - 1s - 22ms/step - loss: 4.0881e-04
Epoch 11/100
25/25 - 0s - 13ms/step - loss: 3.7897e-04
Epoch 12/100
```

```
25/25 - 0s - 10ms/step - loss: 4.5805e-04
Epoch 13/100
25/25 - 0s - 8ms/step - loss: 6.0975e-04
Epoch 14/100
25/25 - 0s - 6ms/step - loss: 0.0013
```

```
# Plotting
plt.figure(figsize=(12, 6))
plt.plot(years, forest_area, label='Observed Forest Area')
plt.plot(future_years, predictions, label='LSTM Forecast', marker='x', linestyle='--', color='r', markeredgecolor='black',ms='10')

plt.xlabel('Year')
plt.ylabel('Forest Area')
plt.title('Forest Area in India (LSTM Forecast)')
plt.legend()
plt.grid(True)
plt.show()

# Print predictions
for year, area in zip(future_years, predictions):
    print(f"Forecasted Forest Area for {year}: {area[0]:.2f}")
```

[17]




```
Forecasted Forest Area for 2023: 72853.39
Forecasted Forest Area for 2024: 73167.20
Forecasted Forest Area for 2025: 73483.06
Forecasted Forest Area for 2026: 73809.36
Forecasted Forest Area for 2027: 74154.13
Forecasted Forest Area for 2030: 74525.78
Forecasted Forest Area for 2040: 74962.31
Forecasted Forest Area for 2050: 75431.69
```

[18] `model.summary()`

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 5, 50)	10,400
lstm_1 (LSTM)	(None, 50)	20,200
dense (Dense)	(None, 1)	51

Total params: 91,955 (359.20 KB)
Trainable params: 30,651 (119.73 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 61,304 (239.47 KB)

10.1 PLAGIARISM REPORT

11. REFERENCES

Journal References:

[1] Rolnick, D., Donti, P., Kaack, L., Kochanski, K., Lacoste, A., Sankaran, K., Ross, A., Milojevic-Dupont, N., Jaques, N., Waldman-Brown, A., Luccioni, A., Maharaj, T., Sherwin, E., Mukkavilli, S., Körding, K., Gomes, C., Ng, A., Hassabis, D., Platt, J., & Creutzig, F. (n.d.). ***Tackling Climate Change with Machine Learning.***

https://arxiv.org/pdf/1906.05433

*[2] Kumar, Sachin. "**A Novel Hybrid Machine Learning Model for Prediction of CO2** Using Socio-Economic and Energy Attributes for Climate Change Monitoring and Mitigation Policies." Ecological Informatics, vol. 77, 1 Nov. 2023, pp. 102253–102253*

https://doi.org/10.1016/j.ecoinf.2023.102253.

[3] ***Irvin****, Jeremy, et al. "**ForestNet: Classifying Drivers of Deforestation** in Indonesia Using Deep Learning on Satellite Imagery." ArXiv (Cornell University), 10 Nov. 2020*

https://doi.org/10.48550/arxiv.2011.05479.

*[4] Tur, Rifat, et al. "**Sea Level Prediction Using Machine Learning.**" Water, vol. 13, no. 24, 13 Dec. 2021, p. 3566*

https://doi.org/10.3390/w13243566.

*[5] Grant Buster, Jordan Cox, Brandon N. Benton, and Ryan N. King "**Tackling extreme urban heat: a machine learning approach** to assess the impacts of climate change and the efficacy of climate adaptation strategies in urban microclimates "*

https://arxiv.org/pdf/2411.05952

Web References:

Global Conference Summit Paper (NOV 2024) : **COP 29**

- *Summary of Global Climate Action at COP 29*. (n.d.). https://unfccc.int/sites/default/files/resource/Summary_Global_Climate_Action_at_COP_29.pdf

International Centre for Education and Research (ICER), VIT-Bangalore

PROJECT
STUDENT'S WORKLOG SHEET

Student Name: SHIVAM
Register Number: 24MSP3080
Project Title: Prediction of Climate Change and its Impact on the World.
Domain: ENVIRONMENT
Tools Learned / used:

Day	Date	Task	Student's Remarks	Signature of Student with Date	Signature of the Project Mentor
Day	25.11.2024	Domain Selection Base Paper Selection, Finalising the Title and Problem statement.	TITLE, DATASET FINALISING BASE PAPER + 4	25-11-24	25/11/24

Day	27-11-24	Final Abstract (Objective, Dataset, methodology, Tools, Expected Outcome) and Review of Related Literature	ABSTRACT METHODOLOGY REVIEW OF LITERATURE DONE	27-11-24	27/11/24
Day	26-11-24	ZEROTH REVIEW	PRESENTED PPT	26-11-24	26/11/24
Day	2-12-24	Approach to the Problem write up 2 pages (Aim, Research Objective, Research questions, Algorithm, Tools, Dataset, proposed architecture, expected outcome, References). Perform EDA	REVIEW-1 DATA CLEANING ARCHITECTURE EDA	2-12-24	2/12/24
Day	3-12-24	Implementation	APPLIED LIN.REG AREMA TO ANNUAL MEAN GLOBAL TEMP DATASET	3-12-24	3/12/24
Day	4-12-24	Implementation	APPLIED LIN.REG, PREDICTION ON CO_2 CONCENTRATION	4-12-24	4/12/24

Day	5-12-24	Implementation	ONLINE CLASS Implementation of Code for Mean Sea Level	Sivan. 5-12-24	5/12/24
Day	9-12-24	Implementation	Implementation of Code for Forest Area DataSet	Sivan 9-12-24	9/12/24
Day	12-12-24	Second Review	Presented 2nd REVIEW	Sivan. 12-12-24	12/12/24
Day	12-12-24	Results and Discussion	Results and Interpretation Shown	Sivan 12-12-24	12/12/24
Day	12-12-24	Conclusion and future scope	Future Scope Discussed with guide	Sivan. 12-12-24	12/12/24

Day	13-12-24	Rough Draft of Report submission	Prepared Rough Draft.	Sivam. 13-12-24	13/12/24
Day	13-12-24	PPT for Final Review Submission	Shown final ppt to Guide	Sivam. 13-12-24	13/12/24
Day	13-12-24	PPT for Final Review Submission and Project Report Submission on approval by Project Mentor	Final ppt and Project Report Approval.	Sivam. 13-12-24	13/12/24
Day	13-12-24	Final Review	Presented final Review ppt to External Examiner	Sivam. 13-12-24	Vanuja 13/12/24