A project report on

EMPOWERING INDIA'S EV FUTURE: TIME SERIES ANALYSIS AND POWERBI VISUALIZATION OF ELECTRIC VEHICLE ADOPTION

Submitted in partial fulfillment for the award of the degree of

Master of Science

In

Business Statistics

by

VRINDA S NAIR
(22MBS0024)

Under the guidance of

Dr. Jitendra Kumar
School of Advanced Sciences
VIT, Vellore

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

MAY, 2024

DECLARATION

I hereby declare that the thesis entitled "EMPOWERING INDIA'S EV FUTURE: TIME SERIES AND POWERBI VISUALIZATION OF ELECTRIC VEHICLE ADOPTION" submitted by me, for the award of the degree of Master of Science in Business Statistics to VIT is a record of bonafide work carried out by me under the supervision of Dr. Jitendra Kumar, Professor, School of Advanced Sciences, Vellore Institute of Technology, Vellore.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Date: 08-05-2024 Signature of the Candidate

Vrinda S Nair

CERTIFICATE

This is to certify that the thesis entitled "EMPOWERING INDIA'S EV FUTURE: TIME SERIES AND POWERBI VISUALIZATION OF ELECTRIC VEHICLE ADOPTION" submitted by Vrinda S Nair (22MBS0024), School of Advanced Sciences, VIT, for the award of the degree of *Master of Science in Business Statistics*, is a record of bonafide work carried out by her under my supervision during the period, 03.01.2024 to 08.05.2024, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfils the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore
Date : 08-05-2024

Signature of the Guide
Department of Mathematics
SAS, VIT- Vellore

Signature of External Examiner

Head, Department of Mathematics
Dr. JAGADEESH KUMAR M.S
SAS, VIT- Vellore

ACKNOWLEDGEMENTS

With immense pleasure and deep sense of gratitude, I wish to express my sincere thanks to my guide Dr. Jitendra Kumar, School of Advanced Sciences, VIT, Vellore without his motivation and continuous encouragement, this research would not have been successfully completed.

I am grateful to the Chancellor of VIT, Vellore, Dr. G. Viswanathan, the Vice Presidents and the Vice Chancellor for motivating me to carry out research in the Vellore Institute of Technology, Vellore and also for providing me with infrastructural facilities and many other resources needed for my research.

I express my sincere thanks to Dr. Arunai Nambi Raj N, Dean, School of Advanced Sciences, VIT, Vellore, for her kind words of support and encouragement. I like to acknowledge the support rendered by my classmates in several ways throughout my research work.

I wish to thank Dr. JAGADEESH KUMAR M.S, Head of the Department of Mathematics, School of Advanced Sciences, VIT, Vellore for his encouragement and support.

I wish to extend my profound sense of gratitude to my parents and friends for all the support they made during my research and also providing me with encouragement whenever required.

Signature of the Student

Vrinda S Nair

ABSTRACT

EV also known as Electronic Vehicle are those vehicles that run entirely or partially on electricity. EVs have become very popular in the previous ten years. Thomas Davenport created the first electric motor that ran on batteries in 1834. Many electric carriages, trams, vehicles, and trains were developed in the late 1800s as a result of this discovery. The first Roadster was released by Tesla in the 2000s. In an effort to bust the myth that electric cars are pricey, several automakers are working hard to build reasonably priced electric automobiles in 2021.

Electric cars are less expensive to operate and have a minimal environmental impact because they use little to no fossil fuels (diesel or gasoline). This is because EVs have fewer moving parts, which means they require less maintenance. Because of their superior energy retention, longer lifespan, and low self-discharge rate of 5% per month, lithium-ion batteries are currently regarded as the industry standard for battery electric vehicles (EVs), even though some EVs run on lead acid or nickel metal hydride batteries. Although attempts have been made to improve the safety of these batteries, problems still exist with them despite their increased efficiency. Their vulnerability to thermal runaway, which has caused explosions and fires in cars, is one of these issues.

The objective of the present work is firstly to visualize the electric vehicle sales in India during the past few years using PowerBI. Secondly, to identify the future trends and sales for the upcoming years and its future impact to the country.

***Keywords*:**

PowerBI, Electric Vehicles (EV), Visualization, Trends, Low-Cost Automobiles.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Objective:

- Transport is a basic need for modern living. Pollutant-producing vehicles such as those running on gasoline or diesel are quickly being replaced by fully electric vehicles. Completely electric cars (EVs) emit no emissions from exhaust and are much better for the environment.
- An electric vehicle has significantly lower running costs than a comparable gasoline or diesel vehicle. It is more economical to charge an electric car rather than fill it up because electric vehicles need less energy to run than vehicles powered by gasoline or diesel.
- Electric vehicles have a lot fewer moving parts than internal combustion vehicles, which means that their maintenance costs are incredibly low. Compared to conventional gasoline or diesel vehicles, maintenance requirements for electric vehicles are lower. Consequently, the annual cost of maintaining an electric vehicle is extremely low.
- One can reduce their carbon footprint by driving an electric vehicle since they don't produce any tailpipe emissions. You can reduce the environmental effect of charging your car even more by choosing renewable energy sources for your home's electrical needs.
- Road tax and registration fees for electric vehicles are less than those for petrol or diesel vehicles. The state in which you reside determines the policies and incentives that the government provides.

1.2 Motivation:

As travelling is playing role in each and every individual, there is a direct effect to the environment. As per the world air quality report, India is ranked **third worst** out of 134 countries in terms of average annual PM2.5 concentration, following Bangladesh and Pakistan (Timesofindia, n.d.). One of the reasons for the major cause of air pollution is emission from vehicles. This can be reduced by various factors such as by using renewable energy, electric vehicles, etc.

Further, there is can huge cost reduction in travelling. A small petrol car will cost you approximately Rs 7-8 per kilometer to operate. It costs between Rs. 1 and Rs. 1.5 per kilometer in an EV. (timesofindia, n.d.)

By 2030, 30 percent of private automobiles, 7 percent of commercial vehicles, and 80 percent of two- and three-wheelers are expected to be electric (timesofindia, n.d.). Thus, my motivation to do this project is to identify the future trends and sales for the upcoming years and its future impact to the country.

1.3 Electric Vehicles

Within an electric vehicle (EV), there is no internal combustion engine to generate power through the burning of fuel and gases. Instead, the vehicle runs on an electric motor. This kind of vehicle is therefore thought to be a viable replacement for cars made today in order to address issues like rising pollution, global warming, the depletion of natural resources, etc. Even though the idea of electric cars has been around for a while, in the last ten years, it has gained a lot of attention due to the negative environmental effects of fuel-powered vehicles, including an increasing carbon footprint (business-standard, n.d.). There has been a drastic rise of EV sales in India from 1.75% in 2021 to 6.38% in the year 2023 which is reflecting the growth of Electric Vehicles for a period of two years (6) (thehindubusinessline, n.d.). There are four types of EVs.

- **Battery Electric Vehicle (BEV):** It does not require an additional source of propulsion and solely uses chemical energy stored in rechargeable battery packs.
- **Hybrid Electric Vehicle (HEV):** Internal combustion engines and battery-powered electric motors power hybrid electric vehicles (HEVs). HEV drivers, recharge their batteries through regenerative braking, in contrast to other electric vehicles.
- **Plug-in Hybrid Electric Vehicle (PHEV):** They are powered by batteries in the case of an electric motor and an internal combustion engine. Since the battery can now hold enough energy to run the electric motor, fuel consumption can be reduced by up to 60%.

- **Fuel Cell Electric Vehicle (FCEV):** Numerous parts of BEVs and FCEVs are identical. FCEVs use fuel cells, which are superior to batteries in many ways, whereas BEVs use battery energy (Ning Wang, 2018).

Advantages

- **Environmentally friendly:** No emissions or petrol exhaust are produced by electric vehicles since they do not burn fuel. Driving an electric car can help you create a cleaner environment because fossil fuel-powered cars are a major contributor to the atmosphere's buildup of hazardous gases..
- **Renewable energy source**: Electric vehicles use renewable energy sources to power themselves, whereas conventional cars burn fossil fuels, depleting the world's supply.
- **Economical:** Compared to fuels like petrol and diesel, which have erratic price fluctuations, electricity is significantly less expensive.
- **Smoother motion and less noise:** It's much more fun to drive an electric vehicle. Due to the absence of any fast-moving parts, they are more silent and produce less noise.
- **Low maintenance:** Due to their smaller number of moving parts, electric car parts deteriorate more slowly than traditional auto parts. Moreover, compared to combustion engines, repairs are less difficult and costly.

Disadvantages

- **High beginning cost:** Many consumers think that electric cars are not as affordable as conventional cars because they are still relatively pricey.
- **Limitations on charging stations:** People who have to travel long distances worry about where to find sufficient charging stations along the way, as they are not always present.
- **Charging takes time:** Charging an electric vehicle takes several hours, as opposed to a few minutes for a conventional car.

- **Reduced driving range:** The maximum distance an electric vehicle (EV) can cover on a single battery charge is known as its driving range. Electric vehicles have a reduced driving range in comparison to conventional cars.
- **Fewer models:** When it comes to looks, style, and customised options, there aren't many electric car models available at the moment.

EVs will eventually outlive gas-powered cars, even though manufacturers of electric vehicles still need to remove the obstacles that currently prevent consumers from making the purchase. GM and Nissan both announced in January 2021 that they plan to transition to electric vehicles by the 2030s. No doubt, other automakers will inevitably follow suit. (intellipaat, n.d.)

CHAPTER 2: LITERATURE REVIEW

A large amount of research has been done recently to create machine learning models that are efficient at detecting malware. Here is a brief literature review:

A "Global Comparison and Assessment of Incentive Policies for Electric Vehicle Promotion," conducted by Ning Wang, Linhao Tang, and Huizhong Pan (2018). provides valuable insights for policymakers, industry stakeholders, and researchers seeking to accelerate the transition towards sustainable transportation systems. By understanding the strengths and limitations of existing incentive policies and learning from global best practices, countries can develop more effective strategies to promote EV adoption and achieve their environmental and energy objectives. (Ning Wang, 2018)

"Technology Development of Electric Vehicles: A Review" by Xiaoli Sun, Zhengguo Li, Xiaolin Wang and Chengjiang Li (2019) provides valuable insights into the current state of EV technology development and identifies opportunities for future innovation and research. By addressing challenges related to battery technology, electric drivetrains, and charging infrastructure, the EV industry can accelerate the transition towards sustainable transportation and contribute to global efforts to mitigate climate change and reduce air pollution. (Xiaoli Sun, 2019)

"Electric Drive Technology: Trends, Challenges, and Opportunities for Future Electric Vehicles" by Iqbal Husain, MD Sariful Islam, Emre Gurpinar, Wensong Yu, Lincoln Xue, Dhrubo Rahman, and Raj Sahu (2020) provides a comprehensive overview of the current state and future prospects of electric drive technology in the context of EVs. By addressing technical challenges and embracing emerging opportunities, the electric drive technology sector can play a pivotal role in advancing the transition towards sustainable transportation and shaping the future of mobility. (Iqbal Husain, 2020)

“Adoption of Electric Vehicles: A Literature Review and Prospects for Sustainability” by Rajeev Ranjan Kumar, Kumar Alok (2019) provides valuable insights into the current state and future prospects of EV adoption in the context of sustainability. By addressing challenges and leveraging opportunities, policymakers, industry stakeholders, and researchers can work together to accelerate the transition towards a sustainable transportation system powered by electric vehicles. (Rajeev Ranjan Kumar, 2019)

“A Review on Electric Vehicles: Technologies and Challenges” by Julio A. Sanguesa 1, Vicente Torres-Sanz, Piedad Garrido, Francisco J. Martinez and Johann M. Marquez-Barja (2021) provides valuable insights into the current state and future prospects of EV technology. By addressing technical, economic, and policy challenges, stakeholders can work together to accelerate the transition towards a sustainable transportation system powered by electric vehicles. (Julio A. Sanguesa 1, 2021)

"Forecasting Charging Demand of Electric Vehicles Using Time-Series Models" by Yunsun Kim and Sahm Kim (2021) offers valuable insights into the challenges and opportunities associated with EV charging demand forecasting. The study demonstrates the effectiveness of time-series models, particularly SARIMA and LSTM, in predicting EV charging behavior with high accuracy and reliability. By leveraging advanced forecasting techniques, stakeholders in the electric vehicle industry can make informed decisions to support the widespread adoption of EVs and build sustainable charging infrastructure for the future. (Kim, 2021)

“Time Series Forecasting using LSTM and ARIMA” by Khulood Albeladi, Bassam Zafar, Ahmed Mueen (2023) might compare the performance of these models on different types of time series data, considering factors such as accuracy, computational complexity, and interpretability. Their study may provide insights into the strengths and limitations of each approach, helping practitioners choose the most suitable method for their forecasting tasks. (Khulood Albeladi, 2023)

In their study, "Investigating the Power of LSTM-Based Models in Solar Energy Forecasting," Nur Liyana Mohd Jailani, Jeeva Kumaran Dhanasegaran, Gamal Alkawsi, Ammar Ahmed Alkahtani, Chen Chai Phing, Yahia Baashar, Luiz Fernando Capretz, Ali Q. Al-Shetwi, and Sieh Kiong Tiong (2023) focus on utilizing LSTM-based models for forecasting solar energy production. Solar energy forecasting is crucial for effective energy management and grid integration, as it helps anticipate fluctuations in solar power generation. (Nur Liyana Mohd Jailani, 2023)

In their research titled "Forecasting Cryptocurrency Prices Using LSTM, GRU, and Bi-Directional LSTM: A Deep Learning Approach," Phumudzo Lloyd Seabe, Claude Rodrigue Bambe Moutsinga, and Edson Pindza (2023) involves collecting historical cryptocurrency price data and implementing Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Bi-Directional LSTM neural networks. These architectures are well-suited for sequential data analysis and are capable of capturing complex patterns and dependencies within the cryptocurrency price time series. (Phumudzo Lloyd Seabe, 2023)

The collective works offer comprehensive insights into various aspects of sustainable transportation and clean energy technologies. Together, they contribute to understanding and advancing the transition to cleaner and more efficient transportation systems.

CHAPTER 3: METHODOLOGY

3.1 Data

The data is collected from the government website. This website consists of sale of Electric Vehicles and I have considered the dataset from 2017 to 2023.

The dataset collected has 85 rows and 35 columns. Since, we are planning to conduct the analysis on the jupyter notebook, we first needed to import the necessary libraries and the dataset to the jupyter notebook.

3.2 Basic Data Analysis

When conducting a basic data analysis, we look for the dataset's fundamental characteristics, such as its size, the datatype of its columns, and whether any of the data are repeating. We will attempt to address any duplicate values that are discovered during this phase.

Additionally, we look for any null values in the dataset. We attempt to determine the percentage of null values in the dataset if any are found. However, at this stage, we won't deal with the dataset's null values. In addition, we'll see if the dataset directly provides the dependent variable, also known as the target variable. We will determine whether the information is numerical or categorical if it is provided directly. We must ensure that the target variable has been encoded to a numerical value if it is categorical. It will make procedures in the future easier. In the event that the target variable is not given explicitly, we will define it using the dataset that is given. Finally, at this stage, we will choose between using a classification model and a regression model.

3.3 Machine Learning

In machine learning, algorithms are developed without the requirement for explicit programming for each task by finding hidden patterns in datasets and using the information

to forecast fresh data of similar kinds. It is used in many different applications, such as automated tasks, fraud detection, recommendation engines, image and speech recognition, natural language processing, and more. Drones, robots, and autonomous cars are also powered by machine learning models, which give them increased intelligence and environmental adaptability.

Generating recommendations is a common task for machine learning. Utilising past data, recommender systems—a popular use case for machine learning—offer users customised suggestions. Regarding Netflix, the system makes recommendations for films and TV series to users based on their viewing history, ratings, and other variables like genre preferences. It does this by combining collaborative filtering and content-based filtering. (8) (geeksforgeeks, ml, n.d.)

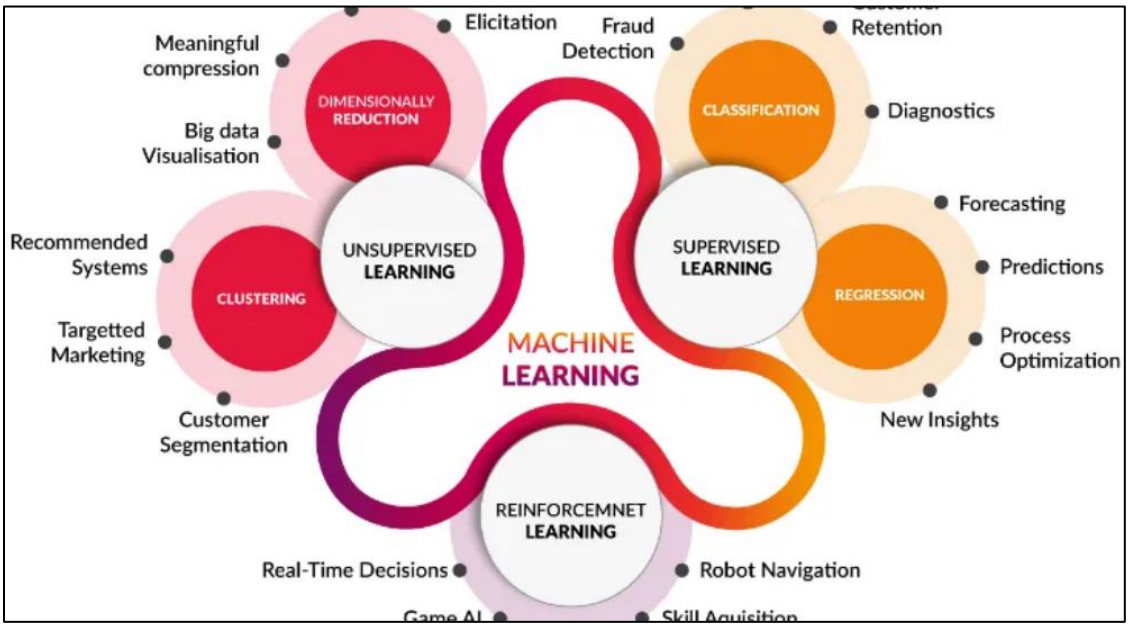


Figure 1: Machine Learning

Machine Learning Model

Machine learning algorithms can be compared to a programme that has been trained to find patterns in fresh data and make predictions. These models are expressed as a mathematical function that receives requests in the form of input data, predicts the data based on the requests, and then outputs the results. These models are fed a set of data to train on before receiving an algorithm to analyse the data, look for patterns in the feed data, and learn from it. After being trained, these models can be utilized to forecast the unseen dataset (javatpoint, n.d.).

Classification of Machine Learning Models

There are three major classifications in Machine Learning Models. They include:

1. Supervised Learning

    With training data as input and a known label or outcome as the output, supervised learning is the easiest model to comprehend. Thus, input-output pairs are the basis for how it operates. It involves building a function that can be trained on a training set of data, which is then applied to unidentified data to produce some predictive performance. Labelled data sets are used to assess task-based supervised learning.

    On simple real-world issues, we can use a supervised learning model. For instance, if we possess a dataset containing data on height and age, we can develop a supervised learning model to anticipate an individual's height based on their age. (javatpoint, n.d.)

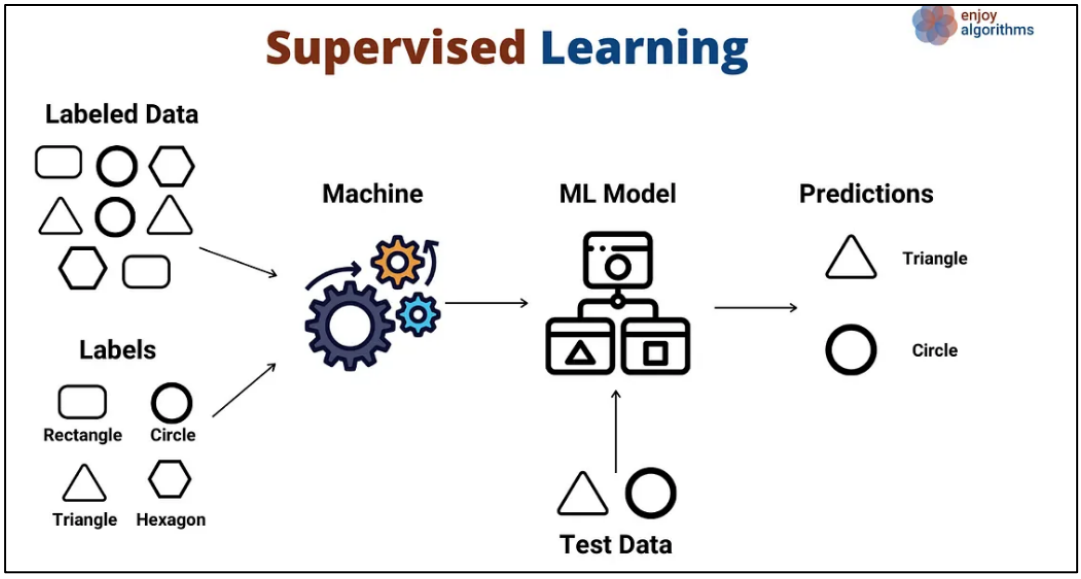


Figure 2: Supervised Learning

They are further classified into two categories:

i. Regression:

Regression analysis is a statistical method used to examine how one or more independent variables (predictor variables) and a dependent variable (target variable) relate to each other. The goal is to identify the best function that best describes the relationship between these variables.

It looks for the model that fits the data the best so that conclusions or predictions can be drawn from it. (10) (Regression, n.d.)

Some of the commonly used Regression models are as follows:

- Linear Regression
- Decision Tree
- Random Forest
- Neural Networks

ii. Classification:

The second kind of supervised learning techniques, known as classification models, are employed to draw inferences from categorical values that are observed. For instance, the classification model can determine whether an email is spam or not, whether a customer will buy the product, etc. In order to divide the output into distinct groups and predict two classes, classification algorithms are utilised.

In classification, the dataset is divided into distinct categories using a classifier model, and each category is given a label (javatpoint, n.d.).

The classification in machine learning is of two types:

a. Binary classification: If there are only 2 possible classes in a problem, then it is called binary classification.
b. Multi-Class Classification: If there are two possible classes in a problem, then it is called multi-class classifier.

Some of the popular classification algorithms are as follows:

- Logistic Regression
- Support Vector Machine
- Naïve Bayes

2. Unsupervised Learning

Unattended In contrast to supervised learning, machine learning models apply the learning process, allowing the model to gain knowledge from unlabelled training datasets. The output is predicted by the model using the unlabelled dataset. Through

the use of unsupervised learning, the model independently discovers hidden patterns in the dataset without any guidance.

Some of the models used in Unsupervised Learning are as follows:

- Clustering
- Association Rule Learning
- Dimensionality Reduction

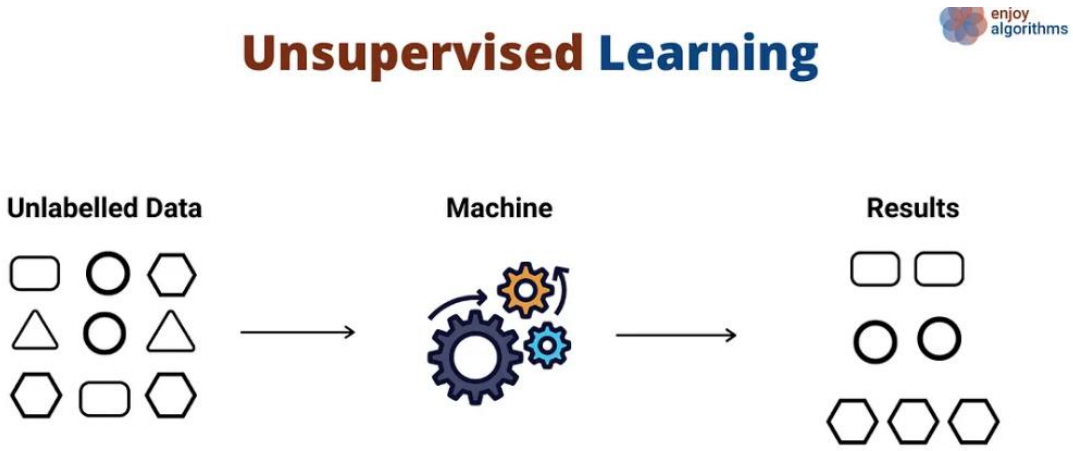


Figure 3: Unsupervised Learning

3. Reinforcement Learning

An algorithm learns actions for a given set of states that lead to a goal state through reinforcement learning. This learning model is feedback-based, meaning that it engages with the surroundings to collect feedback signals after every state or action. To improve their performance, the agent seeks to maximize the positive reinforcement. This feedback serves as a reward system, with each good deed receiving positive feedback and each bad deed receiving negative feedback. The way that the reinforcement learning model behaves is comparable to how people learn because they engage with their surroundings and gain knowledge from experiences.

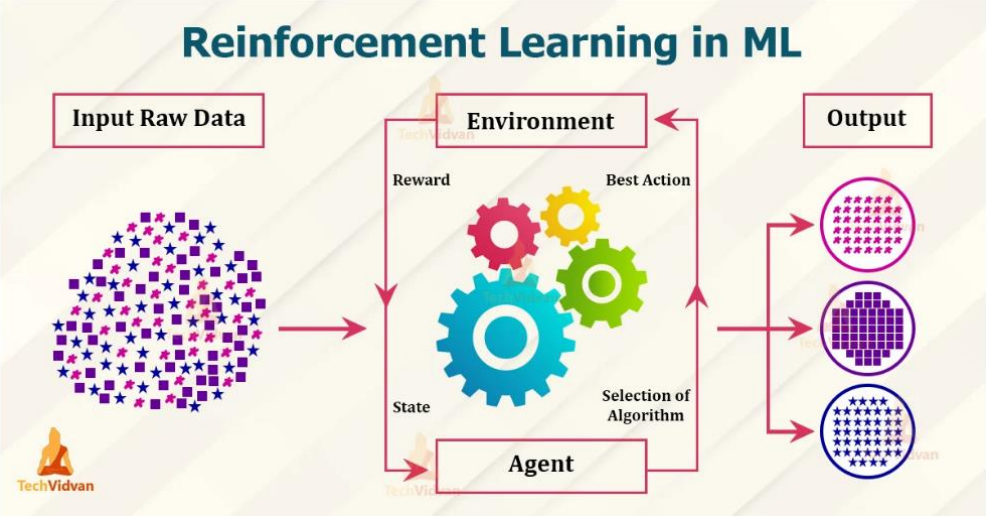


Figure 4: Reinforcement Learning

3.3.1 eXtreme Gradient Boosting (XGBoost)

XGBoost is an optimized distributed gradient boosting library designed to help with the efficient and scalable training of machine learning models. This ensemble learning method combines the predictions of multiple weak models into a single, stronger prediction. "XGBoost," or "Extreme Gradient Boosting," is a machine learning algorithm that has become extremely popular and widely used due to its exceptional performance in a variety of machine learning tasks, such as regression and classification, as well as its ability to handle large datasets.

Because XGBoost can handle missing values effectively, it's a useful tool for handling missing values in real-world data. This ability removes the requirement for a lot of pre-processing. In addition, models can be trained on large datasets rapidly because XGBoost by default supports parallel processing (geeksforgeeks, xgboost, n.d.)

3.3.2 Random Forest

One popular machine learning algorithm is Random Forest, which is used in supervised learning techniques. It is applicable to machine learning issues that incorporate regression and classification. The concept of ensemble learning, which is the process of combining multiple classifiers to solve a difficult problem and improve the functionality of the model, forms the basis of it.

As suggested by the term, "Random Forest is a classifier that consists of multiple decision trees on different subsets of the given dataset and uses the average to increase the dataset's predictive accuracy." The random forest makes predictions by aggregating the majority vote of predictions from all decision trees, as opposed to relying solely on one (javatpoint, random forest, n.d.)

Uses

- When training, it takes less time than other algorithms..
- Even for the large dataset, it operates efficiently and produces output predictions with high accuracy.
- It can still be accurate even if a sizable portion of the data is missing. (javatpoint, random forest, n.d.).

3.4 Time Series

A timeline series is a set of data points that appear sequentially over a predetermined period of time. Cross-sectional data, which records a single point in time, can be used to contrast this. Time series analysis is specifically used to analyse a series of data points collected over time. Time series analysts log data points at predefined intervals over a predefined duration of time, as opposed to logging data points sporadically or arbitrarily. However, the analysis of this type involves more than just collecting data over time.

Time series analysis typically requires a high number of data points to ensure consistency and reliability. When you have a big data set, you can be sure that your analysis can discriminate between noisy data and that the sample size is representative. It additionally guarantees that any observed patterns or trends are typical of the average and can account for seasonal variation. The forecasting process, which makes predictions about the future based on the past, is another application for time series data (tableau, time series, n.d.).

3.4.1. Autoregressive Integrated Moving Average (ARIMA)

Combining the moving average model with the differenced autoregressive model yields the Autoregressive Integrated Moving Average (ARIMA) model. The time series is regressed on its own historical data, as demonstrated by the AR component of ARIMA. The MA component of ARIMA states that the forecast error is a linear combination of prior corresponding errors. As required by the ARIMA model approach, stationary data must be obtained. The ARIMA's I component shows how the data values were changed to differing values of d. Using this combination approach, the ARIMA model fits historical data well and can be used to predict future points in a time series.

It incorporates differencing, which makes the data stationary, moving average, which takes into account the linear combination of previous error terms, and autoregression, which shows the correlation between an observation and several lagged observations. ARIMA is a useful tool in domains like finance, economics, and climate science because it can forecast future values by examining historical data and its sequential patterns (sciencedirect, arima, n.d.).

3.4.2. Seasonal Autoregressive Integrated Moving Average (SARIMA)

Seasonal Autoregressive Integrated Moving Average, or SARIMA for short, is a comprehensive and popular time series forecasting model. A development of the

non-seasonal ARIMA model, it was designed to handle information with seasonal patterns. SARIMA is an effective forecasting tool because it captures both short- and long-term dependencies in the data. This amalgamation blends the ideas of seasonal moving average (MA), integrated (I), and autoregressive (AR) models.

Components of SARIMA

- **Seasonal Component:** Seasonality, denoted by the letter "S" in SARIMA, is the concept of recurring patterns in the data. Any regular interval, such as daily, monthly, or yearly, could be used for this. SARIMA's ability to recognise and model the seasonal component is one of its main advantages.
- **Autoregressive (AR) Component:** The autoregressive component, or "AR" in SARIMA, models the relationship between the current data point and its recorded values in the past. The autocorrelation of the data, or how correlated the data is with itself over time, is captured by it.
- **Integrated (I) Component:** SARIMA's "I" stands for differencing, which turns non-stationary data into stationary data. For time series modelling, stationarity is essential. The number of differences needed to attain stationarity is measured by the integrated component.
- **Moving Average (MA) Component:** The moving average component (represented by the letter "MA" in SARIMA) simulates the relationship between the most recent data point and previous forecast errors. In the data, it aids in capturing transient noise (geeksforgeeks, SARIMA, n.d.).

CHAPTER 4: LANGUAGE, SOFTWARE AND CODE SNIPPET

4.1 Python

Python is a high-level, object-oriented, interpreted language with a flexible semantics. Its dynamic typing, dynamic binding, and high-level built-in data structures make it very attractive for use as a scripting or glue language to connect existing components as well as for Rapid Application Development. Python's straightforward, simple-to-learn syntax prioritises readability, which lowers programme maintenance costs. Python's support for modules and packages encourages program modularity and code reuse. Source or binary versions of the Python interpreter and the extensive standard library are freely available for download on all major platforms.

Python's increased productivity makes programmers fall in love with it quite frequently. The cycle of edit-test-debug is extremely fast because there is no compilation step. Programming in Python is simple to debug because segmentation faults are never caused by bugs or incorrect input. Rather, an exception is raised by the interpreter upon detecting an error. If an exception is not caught by the program, the interpreter prints a stack trace. Setting breakpoints, a source level debugger allows you to examine local and global variables, evaluate arbitrary expressions, step through the code one line at a time, and more. Since it is written in Python, the

debugger demonstrates the language's capacity for self-reflection. However, the simplest method of debugging a program is often to add a few print statements.

Libraries Used are:

- TensorFlow: A machine learning framework created by Google that is freely available and extensively utilized for building and improving neural network models.
- Matplotlib: A complete Python visualization toolbox for creating static, animated, and interactive graphics.
- Pandas: a powerful data analysis and manipulation library that offers functions and data structures for dealing with numerical tables and time series.
- Numpy: Large, multi-dimensional arrays and matrices, as well as a broad range of mathematical functions, are supported by this essential Python scientific computing package.
- SciPy: an open-source software ecosystem that includes modules for interpolation, optimization, integration, and other tasks in mathematics, science, and engineering.
- Scikit-learn: Based on NumPy, SciPy, and Matplotlib, this data mining and analysis tools that are simple to use and efficient are provided by the machine learning library.
- Statsmodel: Time-series analysis, linear regression, and other statistical model estimation and testing are supported by this Python module's classes and functions.
- Pmdarima: An additional Python library that offers more tools for working with time-series data and expands on Statsmodels' functionality, such as automatic forecasting and model selection for ARIMA data. (15) (python, n.d.)

4.2 Jupyter Notebook

Jupyter Notebook is an open-source web application that allows you to create and share documents with real-time code, equations, and text. Project Jupyter's staff is in charge of maintaining Jupyter Notebook. In a single interactive environment, users can do interactive code execution, data visualization, and analysis documentation.

From the IPython project, which formerly had its own IPython Notebook project, Jupyter Notebooks are a spin-off project. The programming languages that it supports most commonly are Julia, Python, and R, hence the name Jupyter. Writing Python programs is made possible by the IPython kernel that comes with Jupyter; however, you can also use any of the more than 100 available kernels (realpython, jupyter, n.d.).

Uses:

Jupyter notebooks are useful for various purposes. Users can run specific codes in notebooks, which are interactive computational environments, view the results, and make changes to the code to achieve desired outcomes or to conduct additional research. Because data exploration requires a lot of repetitions, Jupyter notebooks are frequently utilized for this purpose. It is also utilized in other data science processes, like modeling and experimentation with machine learning. It can also be

used to document code samples. Users can run the independent executable code cells in a Jupyter notebook in any order they choose. Code and markdown cells can be alternated throughout documentation (domino, jupyter, n.d.).

4.3 POWERBI

Microsoft Power BI is an app, connector, and software service suite that turns unrelated information into visually catching and interactive insights. Complex data sources like on-premises or cloud-based hybrid data warehouses as well as straightforward ones like Microsoft Excel can be used with Power BI. You can connect to your data sources with ease using Power BI, visualize your results, and publish and share them with everyone.

A local database or an Excel workbook can be connected to with Power BI in a straightforward and quick manner. Large-scale modeling and real-time analytics can be supported by its robust, enterprise-grade design. This indicates that it can be applied in a range of contexts, ranging from the analytics and decision-making engine behind divisions, teams, or entire companies, to a personal report and visualization tool.

Since Power BI is a Microsoft product and integrates with Excel, many of its features will be recognizable to Excel users (monashdatafluency, powerbi, n.d.).

4.4 Code Snippet

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from statsmodels.tsa.stattools import adfuller
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
import seaborn as sns
```

Loading Dataset

In [2]:
```
# Read the CSV file into a DataFrame
file_path =  "C://Users//Vrinda//Desktop//Courses//Capstone//Data//New//2 Wheeler.csv"
df = pd.read_csv(file_path)
```

Dataset Description

In [3]:
```
df.head()
```

Out[3]:

	Date	Andaman & Nicobar Island	Andhra Pradesh	Arunachal Pradesh	Assam	Bihar	Chandigarh	Chhattisgarh	Delhi	DNH and DD	...	Odisha	Puducherry	Punjab	Rajasthan	Sikkim	Tamil Nadu	Tri
0	2017-01-01	0	0	0	0	0	0	6	2	0	...	1	0	2	3	0	0	
1	2017-02-01	0	0	0	1	0	0	5	2	0	...	4	0	1	10	0	3	
2	2017-03-01	0	0	0	2	1	0	7	2	0	...	0	2	4	16	0	2	
3	2017-04-01	0	0	0	1	0	0	4	4	0	...	0	0	0	1	0	4	
4	2017-05-01	0	0	0	1	0	0	0	8	0	...	2	1	2	1	0	5	

5 rows × 35 columns

In [4]:
```
df.tail()
```

Out[4]:

	Date	Andaman & Nicobar Island	Andhra Pradesh	Arunachal Pradesh	Assam	Bihar	Chandigarh	Chhattisgarh	Delhi	DNH and DD	...	Odisha	Puducherry	Punjab	Rajasthan	Sikkim	Tamil Nadu	Tr
79	2023-08-01	0	1568	0	147	713	125	2088	2116	6	...	2737	197	642	4313	0	6246	
80	2023-09-01	0	1388	0	257	900	134	2178	2002	12	...	3049	136	676	4336	0	5178	
81	2023-10-01	1	1757	0	245	820	157	2195	2611	4	...	3972	197	760	4906	0	6507	
82	2023-11-01	3	1997	0	244	1765	241	3880	3976	15	...	4345	194	1184	7743	0	7096	
83	2023-12-01	7	2372	2	204	1354	137	1865	5380	13	...	4383	224	951	4676	0	5403	

5 rows × 35 columns

In [5]:
```
df.shape
```

Out[5]: (84, 35)

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84 entries, 0 to 83
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Date                      84 non-null     object
 1   Andaman & Nicobar Island  84 non-null     int64
 2   Andhra Pradesh            84 non-null     int64
 3   Arunachal Pradesh         84 non-null     int64
 4   Assam                     84 non-null     int64
 5   Bihar                     84 non-null     int64
 6   Chandigarh                84 non-null     int64
 7   Chhattisgarh              84 non-null     int64
 8   Delhi                     84 non-null     int64
 9   DNH and DD                84 non-null     int64
 10  Goa                       84 non-null     int64
 11  Gujarat                   84 non-null     int64
 12  Haryana                   84 non-null     int64
 13  Himachal Pradesh          84 non-null     int64
 14  Jammu and Kashmir         84 non-null     int64
 15  Jharkhand                 84 non-null     int64
 16  Karnataka                 84 non-null     int64
 17  Kerala                    84 non-null     int64
 18  Ladakh                    84 non-null     int64
 19  Madhya Pradesh            84 non-null     int64
 20  Maharashtra               84 non-null     int64
 21  Manipur                   84 non-null     int64
 22  Meghalaya                 84 non-null     int64
 23  Mizoram                   84 non-null     int64
 24  Nagaland                  84 non-null     int64
 25  Odisha                    84 non-null     int64
 26  Puducherry                84 non-null     int64
 27  Punjab                    84 non-null     int64
 28  Rajasthan                 84 non-null     int64
 29  Sikkim                    84 non-null     int64
 30  Tamil Nadu                84 non-null     int64
 31  Tripura                   84 non-null     int64
 32  Uttar Pradesh             84 non-null     int64
 33  Uttarakhand               84 non-null     int64
 34  West Bengal               84 non-null     int64
dtypes: int64(34), object(1)
memory usage: 23.1+ KB
```

In [7]:
```
df.describe()
```

Out[7]:

	Andaman & Nicobar Island	Andhra Pradesh	Arunachal Pradesh	Assam	Bihar	Chandigarh	Chhattisgarh	Delhi	DNH and DD	Goa	...	Odisha	Puducherry
count	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.000000	84.00000	...	84.000000	84.000000
mean	0.345238	795.928571	0.107143	58.892857	319.285714	39.000000	549.940476	978.845238	2.678571	170.00000	...	850.773810	53.619048
std	1.500956	1085.506220	0.380800	84.457021	448.327510	69.459531	852.791905	1412.246587	4.127956	269.78041	...	1259.416525	78.667809
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	...	0.000000	0.000000
25%	0.000000	54.750000	0.000000	1.750000	19.000000	0.000000	24.000000	13.500000	0.000000	1.00000	...	40.000000	2.750000
50%	0.000000	157.000000	0.000000	8.000000	64.000000	3.000000	81.000000	111.500000	1.000000	5.00000	...	88.500000	12.000000
75%	0.000000	1433.000000	0.000000	114.000000	501.250000	55.750000	865.250000	1840.750000	4.000000	276.00000	...	1698.250000	82.750000
max	10.000000	4370.000000	2.000000	282.000000	1765.000000	362.000000	3880.000000	5380.000000	19.000000	977.00000	...	4383.000000	382.000000

8 rows × 34 columns

In [8]:
```
print(df.columns)
```

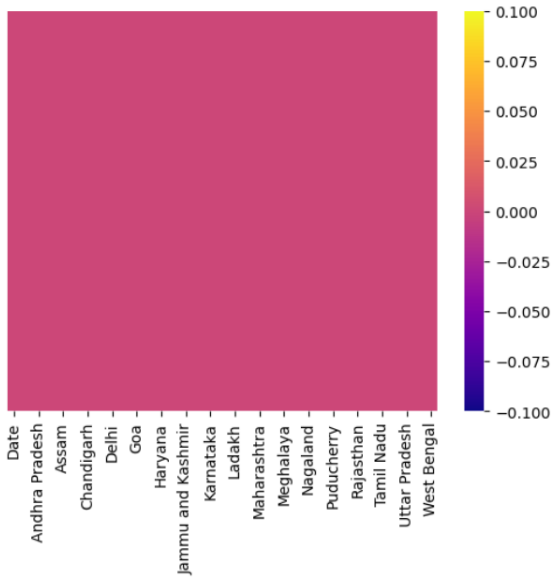
```
Index(['Date', 'Andaman & Nicobar Island', 'Andhra Pradesh',
       'Arunachal Pradesh', 'Assam', 'Bihar', 'Chandigarh', 'Chhattisgarh',
       'Delhi', 'DNH and DD', 'Goa', 'Gujarat', 'Haryana', 'Himachal Pradesh',
       'Jammu and Kashmir', 'Jharkhand', 'Karnataka', 'Kerala', 'Ladakh',
       'Madhya Pradesh', 'Maharashtra', 'Manipur', 'Meghalaya', 'Mizoram',
       'Nagaland', 'Odisha', 'Puducherry', 'Punjab', 'Rajasthan', 'Sikkim',
       'Tamil Nadu', 'Tripura', 'Uttar Pradesh', 'Uttarakhand', 'West Bengal'],
      dtype='object')
```

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Date                        0
Andaman & Nicobar Island    0
Andhra Pradesh              0
Arunachal Pradesh           0
Assam                       0
Bihar                       0
Chandigarh                  0
Chhattisgarh                0
Delhi                       0
DNH and DD                  0
Goa                         0
Gujarat                     0
Haryana                     0
Himachal Pradesh            0
Jammu and Kashmir           0
Jharkhand                   0
Karnataka                   0
Kerala                      0
Ladakh                      0
Madhya Pradesh              0
Maharashtra                 0
Manipur                     0
Meghalaya                   0
Mizoram                     0
Nagaland                    0
Odisha                      0
Puducherry                  0
Punjab                      0
Rajasthan                   0
Sikkim                      0
Tamil Nadu                  0
Tripura                     0
Uttar Pradesh               0
Uttarakhand                 0
West Bengal                 0
dtype: int64
```

In [10]:
```
sns.heatmap(df.isnull(),yticklabels = False, cmap='plasma')
```

Out[10]: <AxesSubplot:>



ARIMA

Reading of CSV file

In [106]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA

# Read the CSV file into a DataFrame
file_path = "C://Users//Vrinda//Desktop//Courses//Capstone//Data//New//2 Wheeler.csv"
df = pd.read_csv(file_path)
```

Setting the date column as index

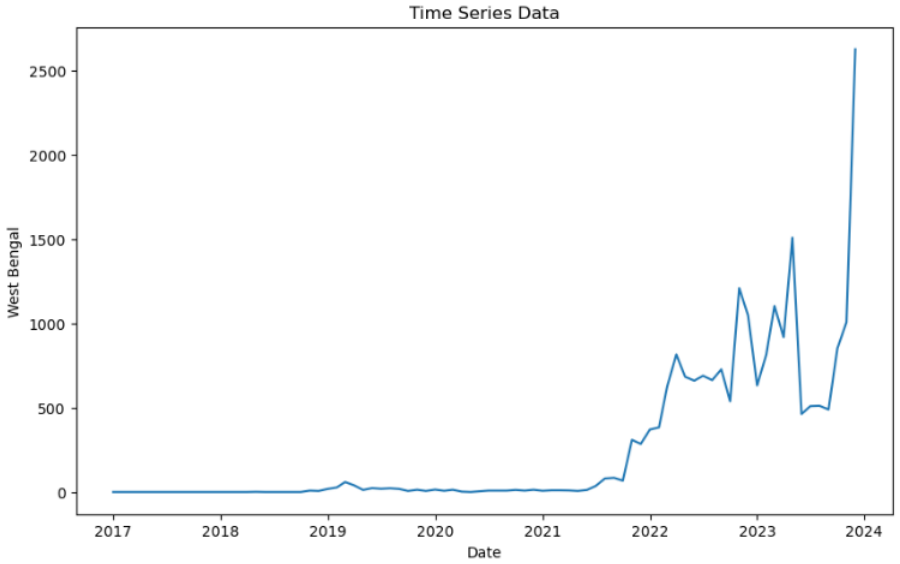
In [107]:
```
# Convert 'Date' column to datetime type
df['Date'] = pd.to_datetime(df['Date'])

# Set 'Date' column as index
df.set_index('Date', inplace=True)
```

Selecting the state

In [108]:
```
# Select the target variable
target = 'West Bengal'
```

Plotting the time series data

In [109]:
```
plt.figure(figsize=(10, 6))
plt.plot(df[target])
plt.title('Time Series Data')
plt.xlabel('Date')
plt.ylabel(target)
plt.show()
```

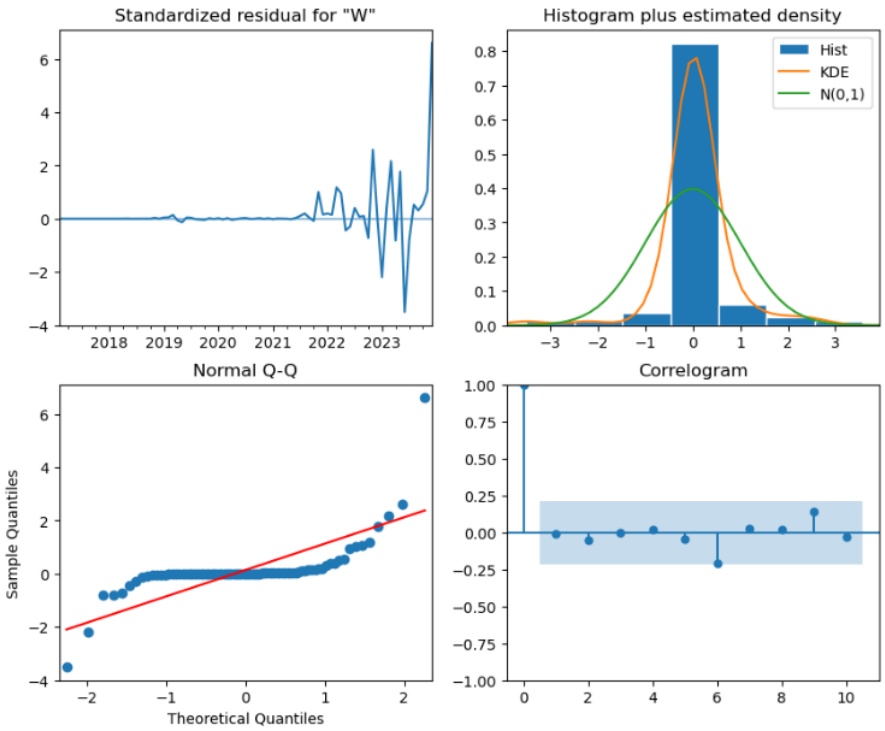


Fitting the ARIMA model

In [110]:
```
# Fit ARIMA model
model = ARIMA(df[target], order=(5,1,0))  # Example order, you can tune this
results = model.fit()

# Print model summary
print(results.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:            West Bengal   No. Observations:                   84
Model:                 ARIMA(5, 1, 0)   Log Likelihood                -573.587
Date:                Sun, 28 Apr 2024   AIC                           1159.173
Time:                        23:57:51   BIC                           1173.686
Sample:                  01-01-2017   HQIC                          1165.004
                       - 12-01-2023
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.2096      0.148     -1.420      0.156      -0.499       0.080
ar.L2          0.1375      0.127      1.079      0.281      -0.112       0.387
ar.L3         -0.0220      0.303     -0.073      0.942      -0.615       0.571
ar.L4         -0.2118      0.192     -1.102      0.270      -0.589       0.165
ar.L5          0.0083      0.189      0.044      0.965      -0.361       0.378
sigma2      5.871e+04   4277.240     13.726      0.000    5.03e+04    6.71e+04
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):              1839.89
Prob(Q):                              0.93   Prob(JB):                         0.00
Heteroskedasticity (H):            1760.90   Skew:                             2.94
Prob(H) (two-sided):                  0.00   Kurtosis:                        25.30
===================================================================================

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

In [111]:
```
# Plot diagnostics of the ARIMA model
results.plot_diagnostics(figsize=(10, 8))
plt.show()
```



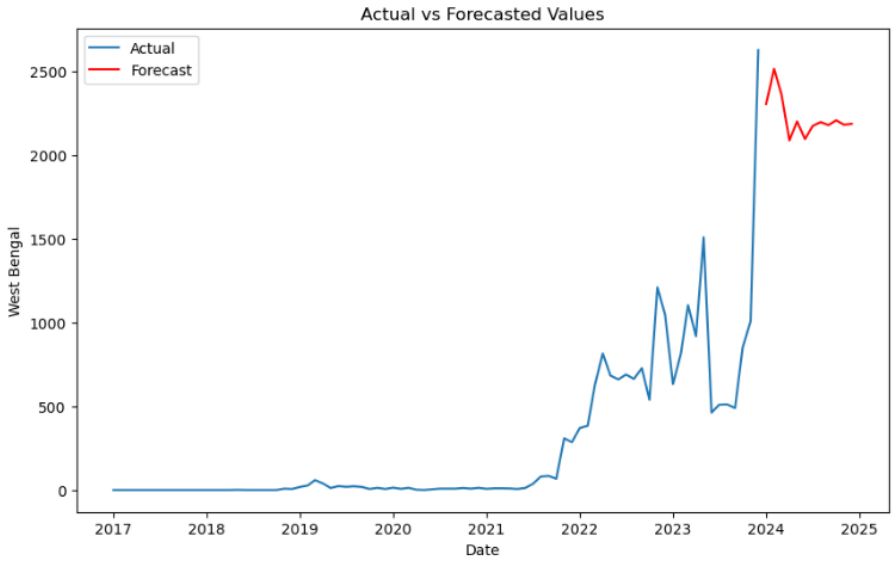
Forecasting of future values

In [112]:
```
forecast_steps = 12  # Number of steps to forecast
forecast = results.forecast(steps=forecast_steps)
# Print forecasted values
print("Forecasted values:")
print(forecast)
```

```
Forecasted values:
2024-01-01    2304.012675
2024-02-01    2513.525696
2024-03-01    2359.755628
2024-04-01    2086.428371
2024-05-01    2199.320313
2024-06-01    2094.420121
2024-07-01    2172.261933
2024-08-01    2195.657847
2024-09-01    2177.598912
2024-10-01    2206.040399
2024-11-01    2179.723561
2024-12-01    2185.236873
Freq: MS, Name: predicted_mean, dtype: float64
```

Plotting of original data along with forcasted values

In [113]:
```
plt.figure(figsize=(10, 6))
plt.plot(df.index, df[target], label='Actual')
plt.plot(pd.date_range(start=df.index[-1], periods=forecast_steps+1, freq='MS')[1:], forecast, label='Forecast', color='red')
plt.title('Actual vs Forecasted Values')
plt.xlabel('Date')
plt.ylabel(target)
plt.legend()
plt.show()
```



Computation of RMSE

In [114]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Extract actual values for the forecast period
actual_values = df[target][-forecast_steps:]

# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(actual_values, forecast)
print("Mean Absolute Error (MAE):", mae)

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(actual_values, forecast)
print("Mean Squared Error (MSE):", mse)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)
print("Root Mean Squared Error (RMSE):", rmse)
```

```
Mean Absolute Error (MAE): 1343.7090487487515
Mean Squared Error (MSE): 1969339.8351926936
Root Mean Squared Error (RMSE): 1403.3316910811548
```

SARIMA

In [115]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pmdarima import auto_arima
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.stattools import adfuller
from sklearn.metrics import r2_score, mean_squared_error

# Read the CSV file into a DataFrame
file_path = "C://Users//Vrinda//Desktop//Courses//Capstone//Data//New//2 Wheeler.csv"
df = pd.read_csv(file_path)
```

In [116]:
```
# Constant to add
constant = 1

# Identify columns where zeros are present (excluding the 'Date' column)
zero_columns = df.columns[1:][df.iloc[:, 1:].eq(0).any()]

# Add constant to places where there are zeros
df[zero_columns] = df[zero_columns].apply(lambda x: x + constant if (x == 0).any() else x)

# Set 'Date' column as index
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
```

In [117]:
```
# Select the target variable 'Andhra Pradesh'
y = df['West Bengal']
```

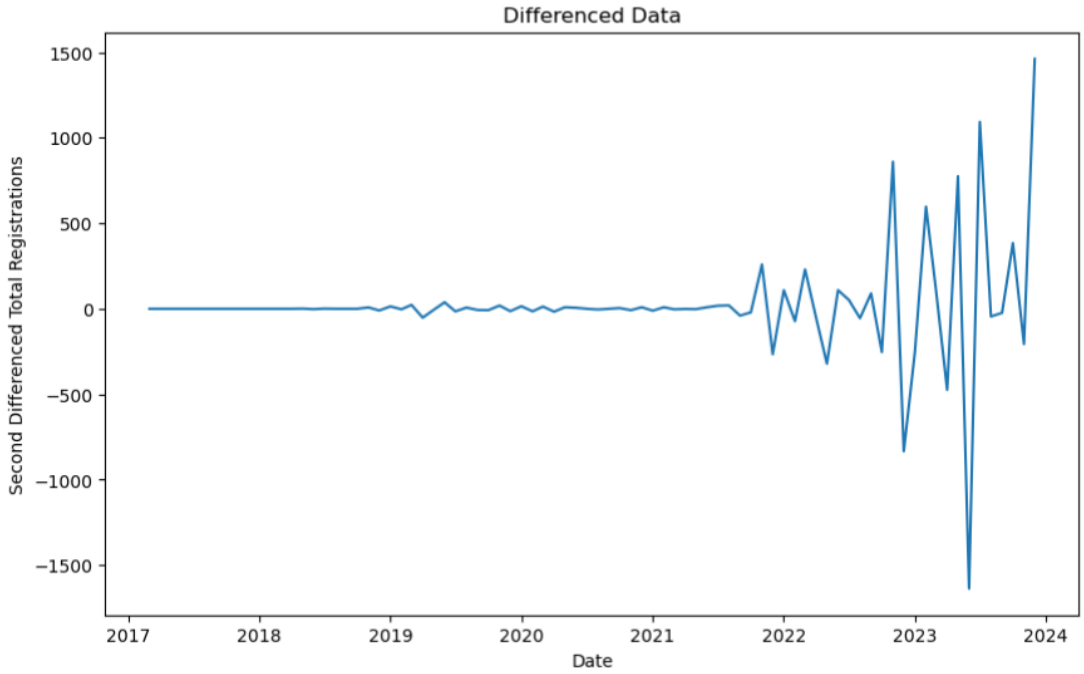
Stationarity Check

In [118]:
```
# Stationarity check
def check_stationarity(y):
    # Perform Dickey-Fuller test
    result = adfuller(y)
    print('ADF Statistic:', result[0])
    print('p-value:', result[1])
    print('Critical Values:')
    for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))

    if result[1] > 0.05:
        print('The series is likely non-stationary.')
    else:
        print('The series is likely stationary.')

# Check stationarity
print('Original Data:')
check_stationarity(y)
```

```
Original Data:
ADF Statistic: 1.67241579880848
p-value: 0.998059355610827
Critical Values:
        1%: -3.525
        5%: -2.903
        10%: -2.589
The series is likely non-stationary.
```

Non-stationary to Stationary

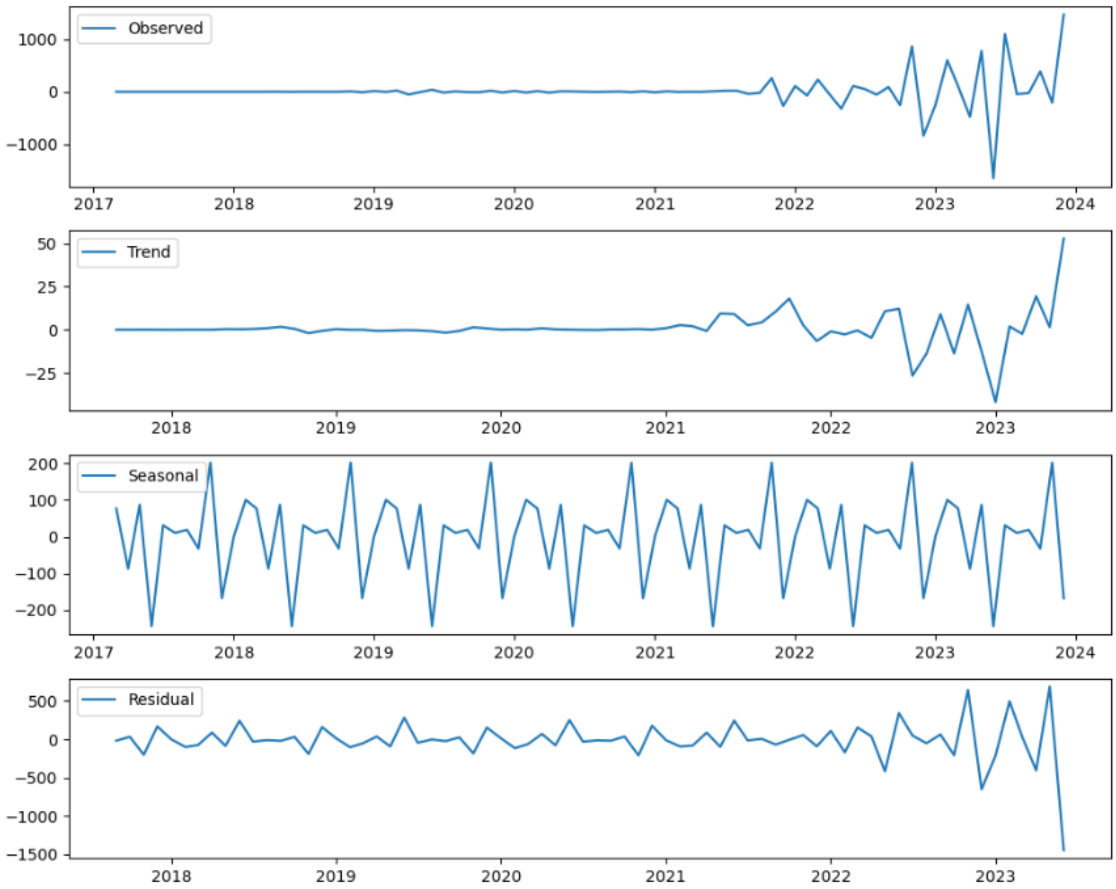
In [119]:
```
# First Differencing
y_diff = y.diff().dropna()

# Second Differencing
y_diff = y_diff.diff().dropna()

# Plot differenced data
plt.figure(figsize=(10, 6))
plt.plot(y_diff)
plt.title('Differenced Data')
plt.xlabel('Date')
plt.ylabel('Second Differenced Total Registrations')
plt.show()

# Check stationarity of differenced data
print('Differenced Data:')
check_stationarity(y_diff)

# Seasonality check using ACF and PACF plots
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```




```
Differenced Data:
ADF Statistic: -3.9329609589686823
p-value: 0.0018052280504689381
Critical Values:
        1%: -3.529
        5%: -2.904
        10%: -2.590
The series is likely stationary.
```

Seasonality Check

In [120]:
```python
from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib.pyplot as plt

# Perform seasonal decomposition on the second differenced data
result = seasonal_decompose(y_diff, model='additive', period=12)  # assuming a seasonal period of 12 months

# Plot the seasonal decomposition
plt.figure(figsize=(10, 8))
plt.subplot(411)
plt.plot(result.observed, label='Observed')
plt.legend()

plt.subplot(412)
plt.plot(result.trend, label='Trend')
plt.legend()

plt.subplot(413)
plt.plot(result.seasonal, label='Seasonal')
plt.legend()

plt.subplot(414)
plt.plot(result.resid, label='Residual')
plt.legend()

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



Fitting the SARIMAX model

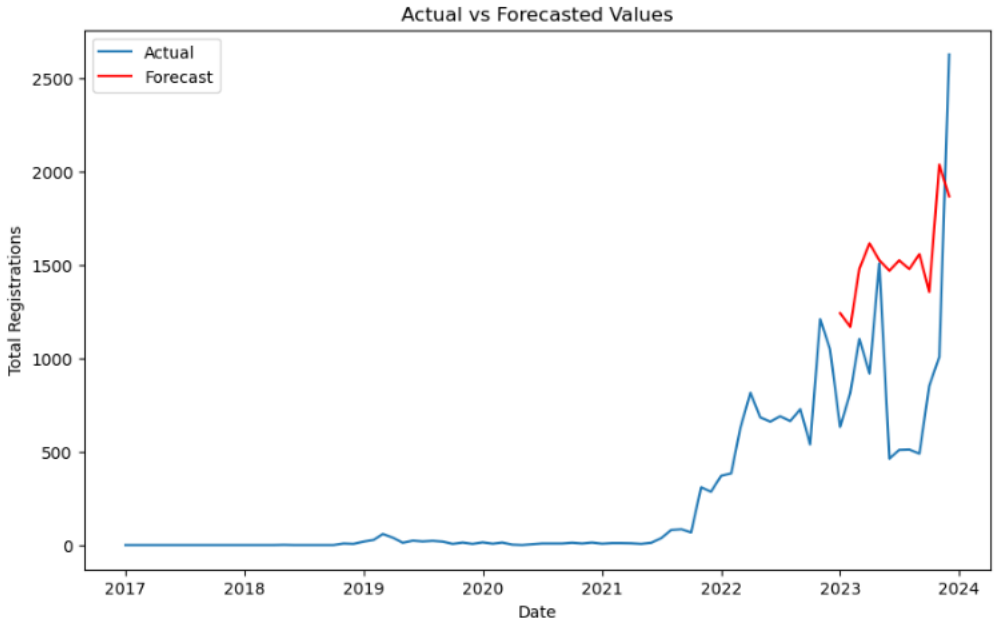
In [121]:
```
# Fit SARIMAX model with auto_arima for parameter selection
model = auto_arima(y, seasonal=True, m=12, trace=True, error_action='ignore', suppress_warnings=True)
order = model.order
seasonal_order = model.seasonal_order

# Print selected model parameters
print("Selected SARIMA model parameters:", order, seasonal_order)

# Fit SARIMAX model
model = SARIMAX(y, order=order, seasonal_order=seasonal_order)
results = model.fit()

# Get summary of SARIMAX model
print(results.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,1,1)[12]             : AIC=inf, Time=1.08 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=1010.894, Time=0.06 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=987.280, Time=0.11 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=997.352, Time=0.22 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=986.190, Time=0.05 sec
 ARIMA(1,1,0)(0,1,1)[12]             : AIC=987.500, Time=0.21 sec
 ARIMA(1,1,0)(1,1,1)[12]             : AIC=989.116, Time=0.31 sec
 ARIMA(2,1,0)(0,1,0)[12]             : AIC=988.162, Time=0.07 sec
 ARIMA(1,1,1)(0,1,0)[12]             : AIC=988.172, Time=0.11 sec
 ARIMA(0,1,1)(0,1,0)[12]             : AIC=995.662, Time=0.06 sec
 ARIMA(2,1,1)(0,1,0)[12]             : AIC=990.159, Time=0.12 sec
 ARIMA(1,1,0)(0,1,0)[12] intercept   : AIC=987.704, Time=0.06 sec

Best model:  ARIMA(1,1,0)(0,1,0)[12]
Total fit time: 2.509 seconds
Selected SARIMA model parameters: (1, 1, 0) (0, 1, 0, 12)
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                         West Bengal   No. Observations:                   84
Model:             SARIMAX(1, 1, 0)x(0, 1, 0, 12)   Log Likelihood                -491.095
Date:                            Mon, 29 Apr 2024   AIC                            986.190
Time:                                    00:06:39   BIC                            990.716
Sample:                                01-01-2017   HQIC                           987.990
                                     - 12-01-2023
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.7882      0.055    -14.216      0.000      -0.897      -0.680
sigma2      5.913e+04   3774.843     15.663      0.000    5.17e+04    6.65e+04
===================================================================================
Ljung-Box (L1) (Q):                   0.03   Jarque-Bera (JB):               609.06
Prob(Q):                              0.87   Prob(JB):                         0.00
Heteroskedasticity (H):             895.35   Skew:                             1.79
Prob(H) (two-sided):                  0.00   Kurtosis:                        16.90
===================================================================================

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

Plotting the Actual vs Forecasted values

In [122]:
```python
# Specify the dates for which you want to forecast
forecast_start_date = max(df.index.min(), pd.to_datetime('2023-01-01'))  # Adjust start date if needed
forecast_end_date = forecast_start_date + pd.DateOffset(months=11)  # Adjust end date if needed

# Perform the forecast
forecast = results.predict(start=forecast_start_date, end=forecast_end_date, dynamic=True)

# Create a DataFrame to store the forecasted values along with dates
forecast_dates = pd.date_range(start=forecast_start_date, end=forecast_end_date, freq='MS')
forecast_df = pd.DataFrame({'Date': forecast_dates, 'Forecasted_Values': forecast})

# Print the forecasted values
print(forecast_df)

# Plot actual vs forecasted values
plt.figure(figsize=(10, 6))
plt.plot(y.index, y, label='Actual')
plt.plot(forecast_dates, forecast, label='Forecast', color='red')
plt.title('Actual vs Forecasted Values')
plt.xlabel('Date')
plt.ylabel('Total Registrations')
plt.legend()
plt.show()
```

```
                Date  Forecasted_Values
2023-01-01 2023-01-01        1241.770247
2023-02-01 2023-02-01        1169.038605
2023-03-01 2023-03-01        1479.611444
2023-04-01 2023-04-01        1615.351195
2023-05-01 2023-05-01        1525.330401
2023-06-01 2023-06-01        1469.242803
2023-07-01 2023-07-01        1524.322123
2023-08-01 2023-08-01        1477.766658
2023-09-01 2023-09-01        1557.968274
2023-10-01 2023-10-01        1356.198320
2023-11-01 2023-11-01        2037.263473
2023-12-01 2023-12-01        1867.330218
```



Computation of Performance Metrics

In [123]:
```
# Calculate residuals
residuals = y - results.fittedvalues

# Calculate Mean Squared Error (MSE)
mse = np.mean(residuals**2)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)

# Calculate R-squared
r2 = r2_score(y, results.fittedvalues)

# Print performance metrics
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
print("R-squared (R2):", r2)
```

```
Mean Squared Error (MSE): 49870.71622159454
Root Mean Squared Error (RMSE): 223.3175233195876
R-squared (R2): 0.744883441408901
```

XGBOOST

In [34]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from xgboost import XGBRegressor
from hyperopt import hp, fmin, tpe, Trials, space_eval, STATUS_OK  # Import STATUS_OK

# Read the CSV file into a DataFrame
file_path = "C://Users//Vrinda//Desktop//Courses//Capstone//Data//New//2 Wheeler.xlsx"
df = pd.read_excel(file_path)
```

In [35]:
```
# Constant to add
constant = 1

# Identify columns where zeros are present (excluding the 'Date' column)
zero_columns = df.columns[1:][df.iloc[:, 1:].eq(0).any()]

# Add constant to places where there are zeros
df[zero_columns] = df[zero_columns].apply(lambda x: x + constant if (x == 0).any() else x)

df['Date'] = pd.to_datetime(df['Date'])
```

Feature Engineering

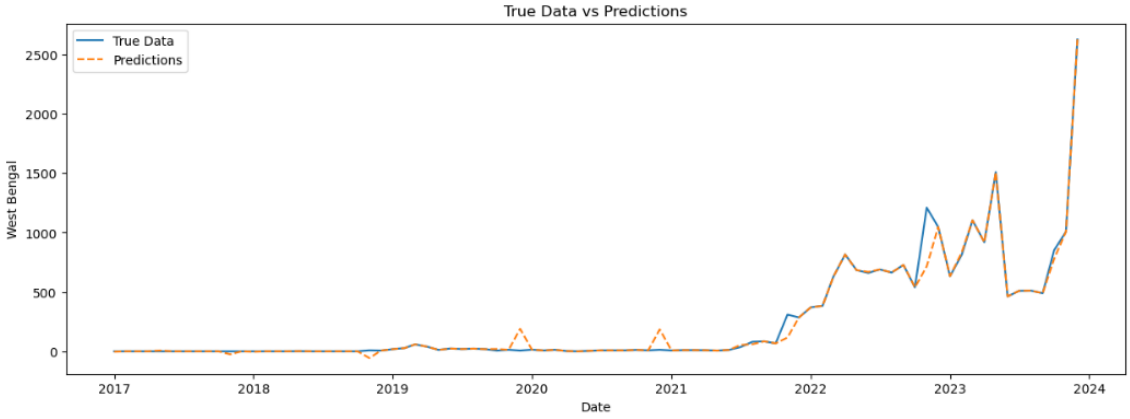
In [85]:
```
# Feature engineering
df['year'] = df['Date'].dt.year
df['month'] = df['Date'].dt.month
df['day'] = df['Date'].dt.day

# Define features and target
features = ['year', 'month', 'day']
target = 'West Bengal'

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(df[features], df[target], test_size=0.2, random_state=42)
```

Fitting the Model

In [37]:
```
# Define the search space for hyperparameters
space = {
    'n_estimators': hp.choice('n_estimators', range(50, 1000, 50)),
    'max_depth': hp.choice('max_depth', range(3, 11)),
    'learning_rate': hp.uniform('learning_rate', 0.01, 0.5),
    'subsample': hp.uniform('subsample', 0.5, 1),
    'colsample_bytree': hp.uniform('colsample_bytree', 0.5, 1),
    'reg_alpha': hp.loguniform('reg_alpha', np.log(0.001), np.log(10)),
    'reg_lambda': hp.loguniform('reg_lambda', np.log(0.001), np.log(10))
}

# Objective function to minimize (in this case, test RMSE)
def objective(params):
    model = XGBRegressor(**params)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    return {'loss': rmse, 'status': STATUS_OK}  # Use STATUS_OK here

# Use Bayesian optimization to find the best hyperparameters
trials = Trials()
best = fmin(fn=objective, space=space, algo=tpe.suggest, max_evals=50, trials=trials)

# Get the best hyperparameters
best_params = space_eval(space, best)
print("Best Hyperparameters:", best_params)

# Train the final model with the best hyperparameters
model = XGBRegressor(**best_params)
model.fit(X_train, y_train)
```

```python
# Make predictions for the next 12 months
last_date = df['Date'].max()
next_dates = pd.date_range(start=last_date + pd.DateOffset(days=1), periods=12, freq='MS')
next_features = pd.DataFrame({
    'year': next_dates.year,
    'month': next_dates.month,
    'day': next_dates.day
})
predictions_next_12_months = model.predict(next_features)

# Evaluate the model
predictions_train = model.predict(X_train)
predictions_test = model.predict(X_test)
train_rmse = np.sqrt(mean_squared_error(y_train, predictions_train))
test_rmse = np.sqrt(mean_squared_error(y_test, predictions_test))
print("Train RMSE:", train_rmse)
print("Test RMSE:", test_rmse)

# Plot predictions
plt.figure(figsize=(15, 5))
plt.plot(df['Date'], df[target], label='True Data')
plt.plot(df['Date'], model.predict(df[features]), label='Predictions', linestyle='--')
plt.xlabel('Date')
plt.ylabel(target)
plt.title('True Data vs Predictions')
plt.legend()
plt.show()
```

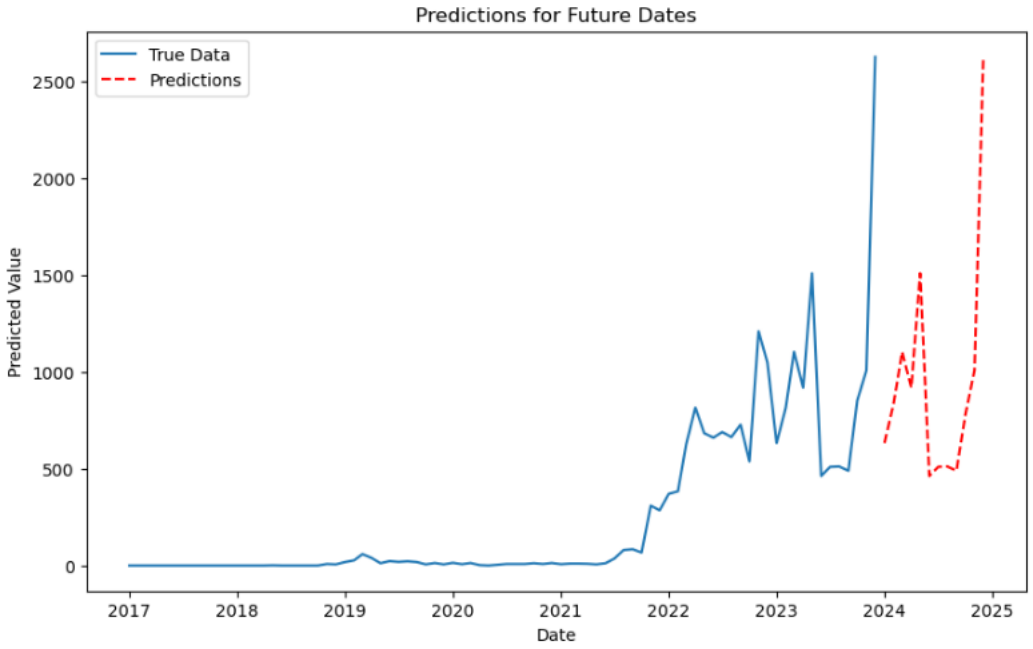
```
100%|██████████████████████████████████████████| 50/50 [00:33<00:00,  1.48trial/s, best loss: 144.65095874797765]
Best Hyperparameters: {'colsample_bytree': 0.9994209380939227, 'learning_rate': 0.24825860293589458, 'max_depth': 7, 'n_estimat
ors': 900, 'reg_alpha': 0.04021913066441679, 'reg_lambda': 0.1784179189655681, 'subsample': 0.7829853225435516}
Train RMSE: 0.019677648373142727
Test RMSE: 144.65095874797765
```



Predictions for the next 12 months

In [41]:
```
# Generate future dates for prediction
start_date = pd.Timestamp('2024-01-01')
end_date = pd.Timestamp('2024-12-31')
future_dates = pd.date_range(start=start_date, end=end_date, freq='MS')

# Create a DataFrame with future dates
future_df = pd.DataFrame({'Date': future_dates})

# Extract year, month, and day from the future dates
future_df['year'] = future_df['Date'].dt.year
future_df['month'] = future_df['Date'].dt.month
future_df['day'] = future_df['Date'].dt.day

# Make predictions for the future dates
predictions = model.predict(future_df[['year', 'month', 'day']])

# Combine predictions with corresponding dates into a DataFrame
predictions_df = pd.DataFrame({'Date': future_dates, 'Predicted_Value': predictions})

# Print or visualize the predictions along with dates
print(predictions_df)

# Plot predictions along with dates
plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df[target], label='True Data')
plt.plot(predictions_df['Date'], predictions_df['Predicted_Value'], label='Predictions', linestyle='--', color='red')
plt.xlabel('Date')
plt.ylabel('Predicted Value')
plt.title('Predictions for Future Dates')
plt.legend()
plt.show()
```

```
         Date  Predicted_Value
0  2024-01-01       633.006531
1  2024-02-01       835.237671
2  2024-03-01      1103.978271
3  2024-04-01       919.030273
4  2024-05-01      1508.989014
5  2024-06-01       462.992218
6  2024-07-01       510.018097
7  2024-08-01       511.989502
8  2024-09-01       490.005432
9  2024-10-01       779.388367
10 2024-11-01      1008.006226
11 2024-12-01      2625.986816
```



In [87]:
```
from sklearn.metrics import r2_score, mean_squared_error

# Calculate RMSE for train and test sets
train_rmse = np.sqrt(mean_squared_error(y_train, predictions_train))
test_rmse = np.sqrt(mean_squared_error(y_test, predictions_test))

# Calculate R2 for train and test sets
train_r2 = r2_score(y_train, predictions_train)
test_r2 = r2_score(y_test, predictions_test)

# Print the performance metrics
print("Train RMSE:", train_rmse)
print("Test RMSE:", test_rmse)
print("Train R-squared (R2):", train_r2)
print("Test R-squared (R2):", test_r2)
```

```
Train RMSE: 0.019677648373142727
Test RMSE: 144.65095874797765
Train R-squared (R2): 0.999999998142201
Test R-squared (R2): 0.8548223168045856
```

RANDOM FOREST

In [89]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Read the CSV file into a DataFrame
file_path = "C://Users//Vrinda//Desktop//Courses//Capstone//Data//New//2 Wheeler.csv"
df = pd.read_csv(file_path)
```

Conversion of sate to datetime format

In [90]:
```
df['Date'] = pd.to_datetime(df['Date'])
```

Feature Engineering

In [91]:
```
# Feature engineering
df['year'] = df['Date'].dt.year
df['month'] = df['Date'].dt.month
df['day'] = df['Date'].dt.day

# Define features and target
features = ['year', 'month', 'day']
target = 'West Bengal'

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(df[features], df[target], test_size=0.2, random_state=42)
```

Fitting the model

In [46]:
```
# Define parameter grid for hyperparameter tuning
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2']
}

# Initialize Random Forest model
model = RandomForestRegressor(random_state=42)

# Perform Grid Search CV
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)
grid_search.fit(X_train, y_train)

# Get best model from Grid Search CV
best_model = grid_search.best_estimator_
```

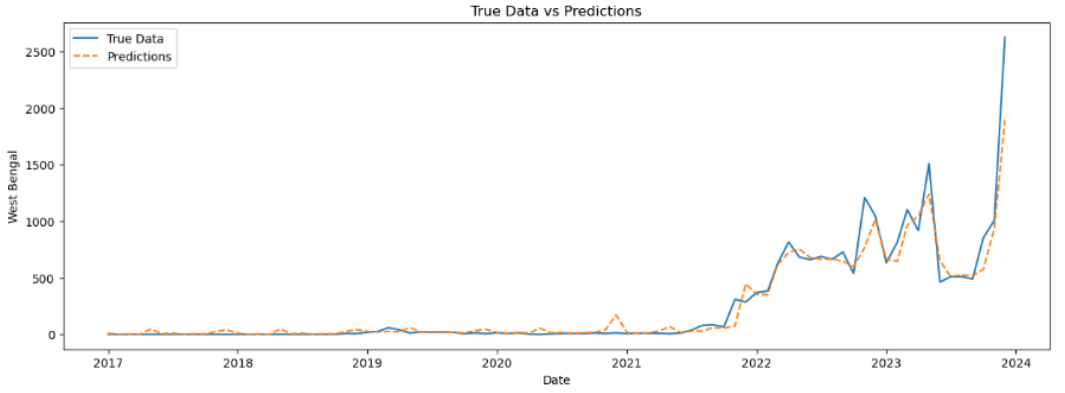
```
Fitting 5 folds for each of 243 candidates, totalling 1215 fits
```

Evaluating the Model ¶

In [93]:
```
# Make predictions
predictions_train = best_model.predict(X_train)
predictions_test = best_model.predict(X_test)

# Evaluate the model
train_rmse = np.sqrt(mean_squared_error(y_train, predictions_train))
test_rmse = np.sqrt(mean_squared_error(y_test, predictions_test))
train_r2 = r2_score(y_train, predictions_train)
test_r2 = r2_score(y_test, predictions_test)

print("Train RMSE:", train_rmse)
print("Test RMSE:", test_rmse)
print("Train R^2:", train_r2)
print("Test R^2:", test_r2)
```

```
Train RMSE: 105.96248379818918
Test RMSE: 151.4333526256908
Train R^2: 0.9461288074465116
Test R^2: 0.8408889640727304
```

In [94]:
```
# Plot predictions
plt.figure(figsize=(15, 5))
plt.plot(df['Date'], df[target], label='True Data')
plt.plot(df['Date'], best_model.predict(df[features]), label='Predictions', linestyle='--')
plt.xlabel('Date')
plt.ylabel(target)
plt.title('True Data vs Predictions')
plt.legend()
plt.show()
```

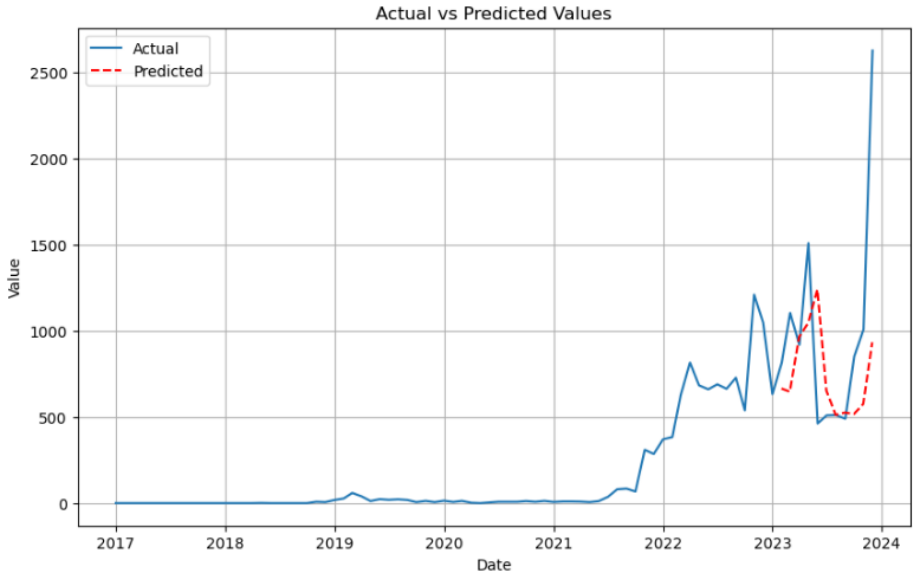


Computation and plotting the Acutal and Prediction values

In [95]:
```
# Let's generate dates from 01-01-2023 to the last date in the dataset
start_date = pd.Timestamp("2023-01-01")
end_date = df['Date'].max()  # Assuming this is the last date in your dataset

dates = pd.date_range(start=start_date, end=end_date, freq='M')

# Extract year, month, and day from the dates
data = {
    'year': dates.year,
    'month': dates.month,
    'day': dates.day
}

# Create a DataFrame with the dates and corresponding features
future_df = pd.DataFrame(data)

# Make predictions for the future dates
predictions_future = best_model.predict(future_df)

# Convert predictions to a DataFrame with corresponding dates
predictions_df = pd.DataFrame({
    'Date': dates,
    'Predicted_Value': predictions_future
})

print(predictions_df)
```

```
         Date  Predicted_Value
0  2023-01-31       664.408333
1  2023-02-28       646.490952
2  2023-03-31       963.479539
3  2023-04-30      1044.249539
4  2023-05-31      1239.039586
5  2023-06-30       652.003475
6  2023-07-31       511.488269
7  2023-08-31       525.068269
8  2023-09-30       516.743825
9  2023-10-31       574.992239
10 2023-11-30       932.545556
```

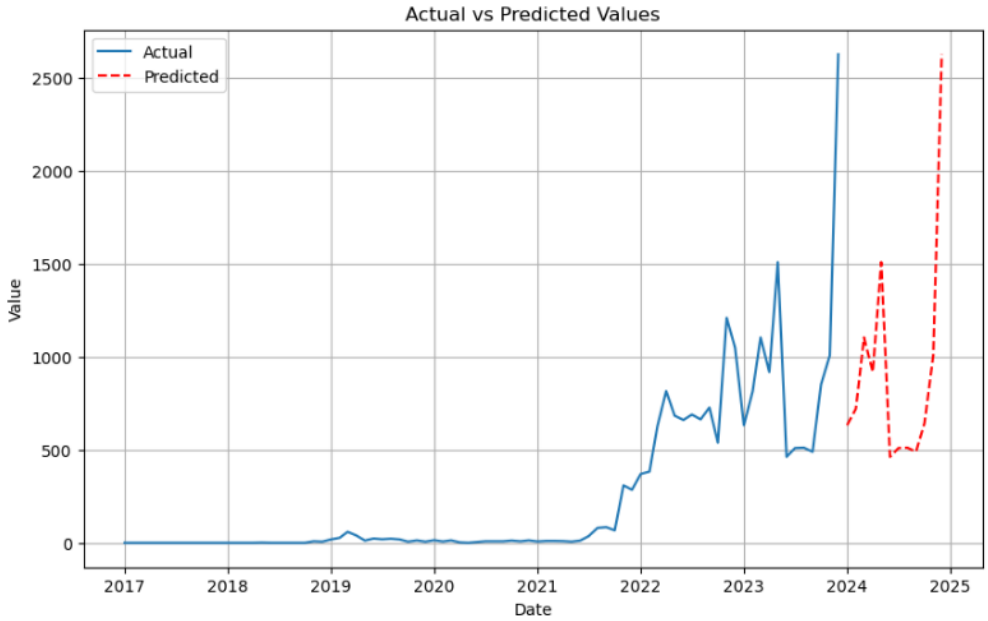
In [96]:
```
plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df[target], label='Actual')
plt.plot(predictions_df['Date'], predictions_df['Predicted_Value'], label='Predicted', linestyle='--', color='red')
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Actual vs Predicted Values')
plt.legend()
plt.grid(True)
plt.show()
```



Future Forecasting

In [52]:
```
# Train the XGBoost model
model = XGBRegressor()
model.fit(X_train, y_train)

# Generate future dates for prediction
start_date = pd.to_datetime('2024-01-01')
end_date = pd.to_datetime('2024-12-01')
future_dates = pd.date_range(start=start_date, end=end_date, freq='MS')

# Extract year, month, and day from future dates
future_data = {
    'year': future_dates.year,
    'month': future_dates.month,
    'day': future_dates.day
}

# Create a DataFrame with future dates and corresponding features
future_df = pd.DataFrame(future_data)

# Make predictions for future dates
predictions = model.predict(future_df)

# Create a DataFrame with future dates and predicted values
predictions_df = pd.DataFrame({'Date': future_dates, 'Predicted_Value': predictions})

# Print and plot the predicted values
print(predictions_df)

plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df[target], label='Actual')
plt.plot(predictions_df['Date'], predictions_df['Predicted_Value'], label='Predicted', linestyle='--', color='red')
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Actual vs Predicted Values')
plt.legend()
plt.grid(True)
plt.show()
```

```
         Date  Predicted_Value
0  2024-01-01       632.005310
1  2024-02-01       719.638428
2  2024-03-01      1102.981567
3  2024-04-01       918.032043
4  2024-05-01      1507.974487
5  2024-06-01       462.003540
6  2024-07-01       509.022522
7  2024-08-01       510.952942
8  2024-09-01       489.031555
9  2024-10-01       643.111694
10 2024-11-01      1006.999512
11 2024-12-01      2624.999756
```



```
In [98]: from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

         # Compute Mean Absolute Error (MAE)
         test_mae = mean_absolute_error(y_test, predictions_test)

         # Compute Mean Squared Error (MSE)
         test_mse = mean_squared_error(y_test, predictions_test)

         # Compute Root Mean Squared Error (RMSE)
         test_rmse = np.sqrt(test_mse)

         # Compute R2 score
         test_r2 = r2_score(y_test, predictions_test)

         # Print the performance metrics
         print("Test MAE:", test_mae)
         print("Test MSE:", test_mse)
         print("Test RMSE:", test_rmse)
         print("Test R2:", test_r2)

         Test MAE: 92.5034221474025
         Test MSE: 22932.060287456807
         Test RMSE: 151.4333526256908
         Test R2: 0.8408889640727304
```

4.5 Visualizations

Visualizations are done using PowerBI. A dashboard has been created in PowerBI desktop for easy analysis and understanding.

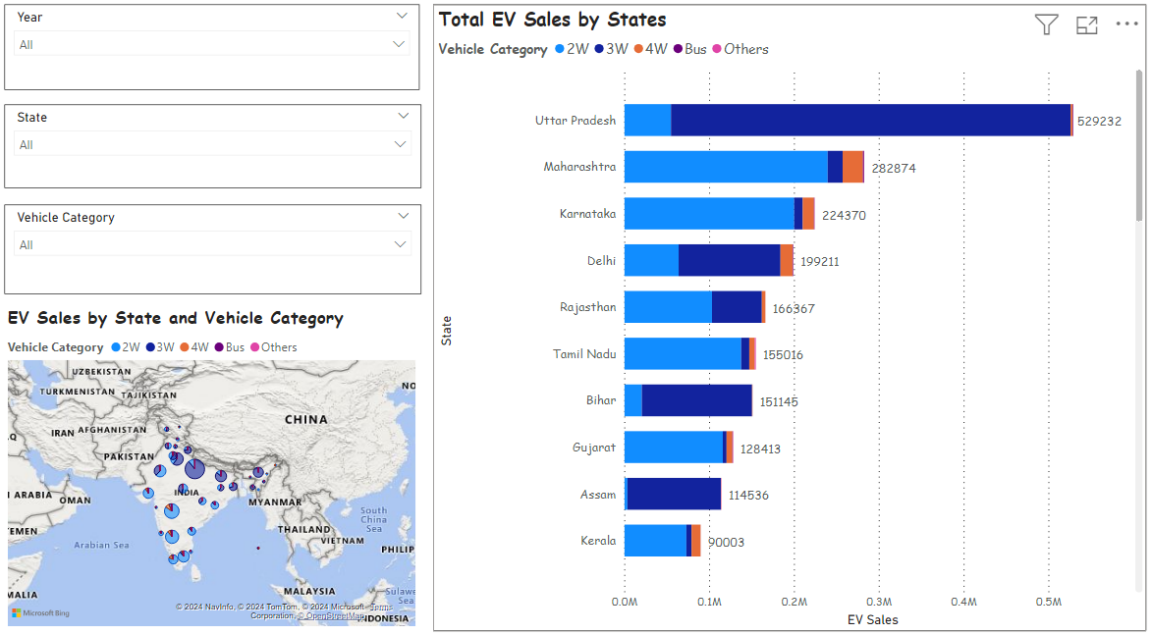


Figure 5: Electric Vehicle Sales in India – State Wise

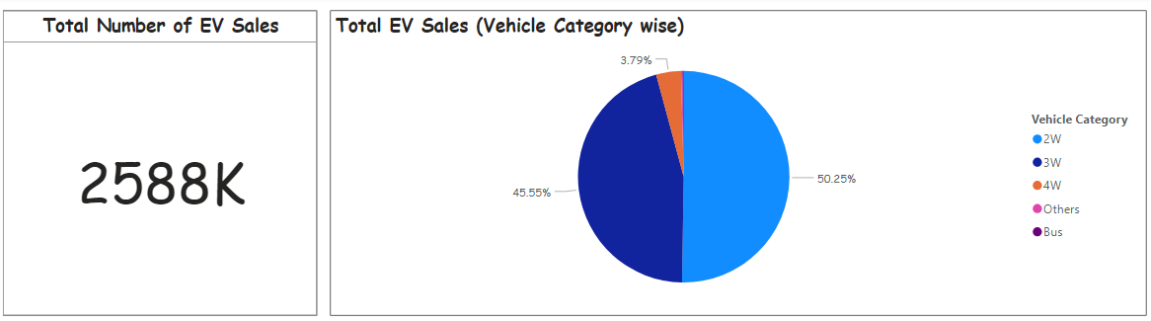


Figure 6: Electric Vehicle Sales in India – Vehicle Category Wise

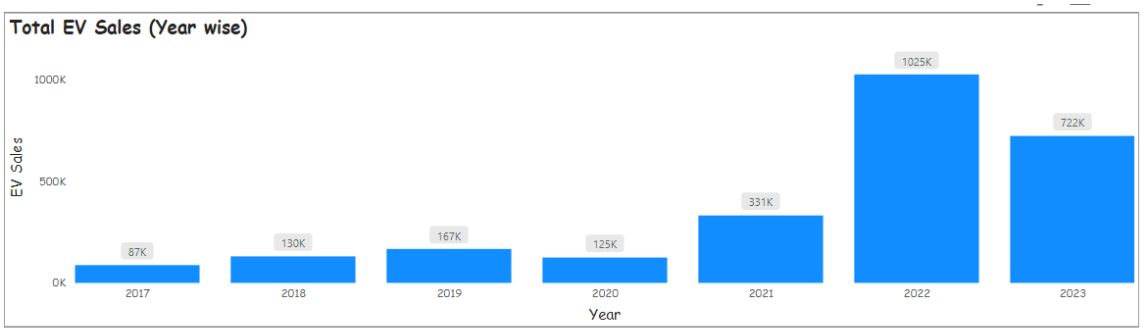


Figure 7: Electric Vehicle Sales in India – Year Wise

From the above visualization, the bar graph shows the sales of Electric Vehicle done from 2017 to 2023 considering all the sales and all types of vehicle category. It also shows geographical plot which shows the sales of EVs. Further, we can filter the required details as per the year, state and vehicle category.

It also shows the total number of EVs sold and also the number of EVs sold by vehicle category. Further, we can analyse the year wise sales done from 2017 to 2023.

CHAPTER 5: RESULTS AND DISCUSSIONS

5.1 Basic Data Analysis

Excel format was used to store the data. Pandas library was therefore used to import the data into the Jupyter notebook. According to the basic data analysis, there were 85 rows and 35 columns. The dataset consists sales of Electric Vehicles in India. It has the number of sales done in every state from the year 2017 to 2023. There were no null nor duplicate values in the dataset. Using the seaborn library and the heatmap provided below, we can attempt to plot the distribution in the dataset:

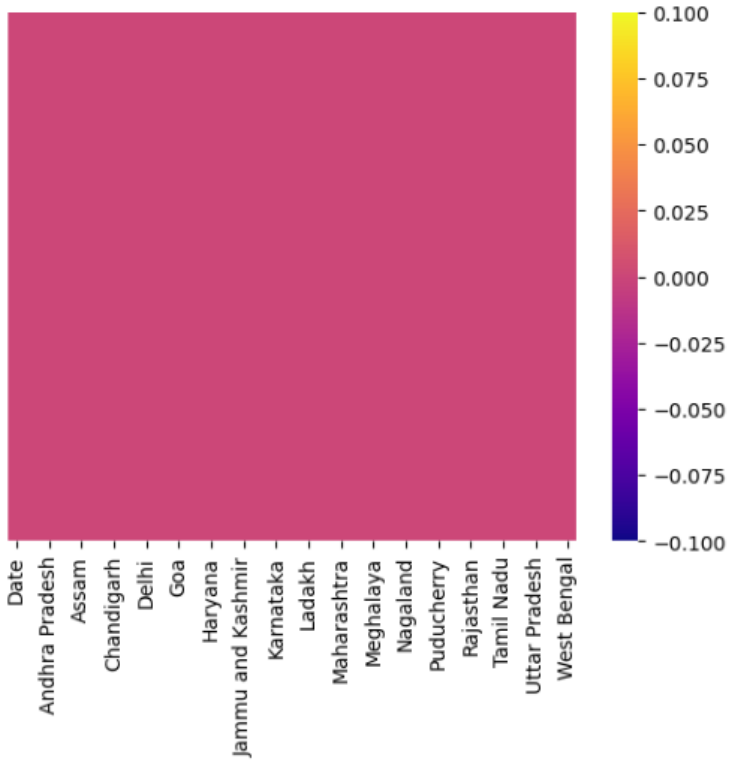


Figure 8: Heat Map

Upon analysis of the dataset using heatmap, it was found that there are no null values in the dataset.

5.2 Time Series Models

1. ARIMA

```
Forecasted values:
2024-01-01    2304.012675
2024-02-01    2513.525696
2024-03-01    2359.755628
2024-04-01    2086.428371
2024-05-01    2199.320313
2024-06-01    2094.420121
2024-07-01    2172.261933
2024-08-01    2195.657847
2024-09-01    2177.598912
2024-10-01    2206.040399
2024-11-01    2179.723561
2024-12-01    2185.236873
Freq: MS, Name: predicted_mean, dtype: float64
```

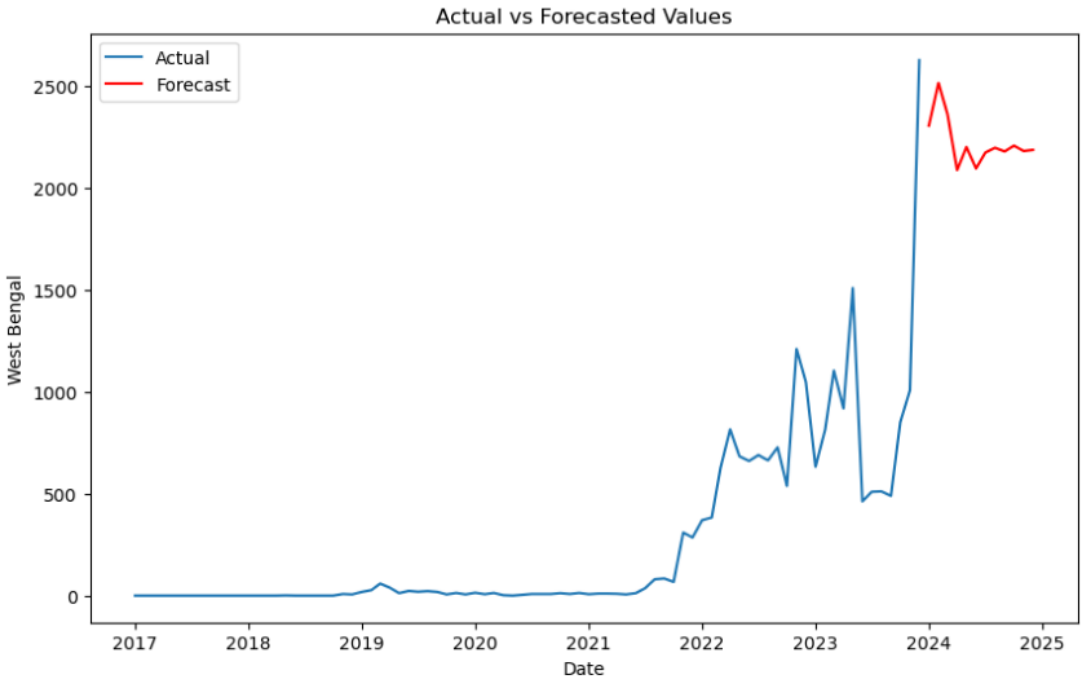


Figure 9: Line Graph on Actual and Forecasted Values – ARIMA

```
Mean Absolute Error (MAE): 9653.40225791565
Mean Squared Error (MSE): 93862310.7776035
Root Mean Squared Error (RMSE): 9688.25633319038
```

Figure 10: Performance Metrics - ARIMA

Figure above displays the predicted values as well as the actual values from the dataset. These forecasted values are from 2024-01-01 to 2024-12-01. It is giving an RMSE of 9688.25 for the state of West Bengal.

2. SARIMA

We checked stationarity of the dataset and made the data from non-stationary to stationary. Also, we checked the seasonality of the dataset and found that the data follows seasonality condition.

```
Best model:  ARIMA(1,1,0)(0,1,0)[12]
Total fit time: 2.509 seconds
Selected SARIMA model parameters: (1, 1, 0) (0, 1, 0, 12)
```

Figure 11: SARIMA parameter

	Date	Forecasted_Values
2023-01-01	2023-01-01	1241.770247
2023-02-01	2023-02-01	1169.038605
2023-03-01	2023-03-01	1479.611444
2023-04-01	2023-04-01	1615.351195
2023-05-01	2023-05-01	1525.330401
2023-06-01	2023-06-01	1469.242803
2023-07-01	2023-07-01	1524.322123
2023-08-01	2023-08-01	1477.766658
2023-09-01	2023-09-01	1557.968274
2023-10-01	2023-10-01	1356.198320
2023-11-01	2023-11-01	2037.263473
2023-12-01	2023-12-01	1867.330218

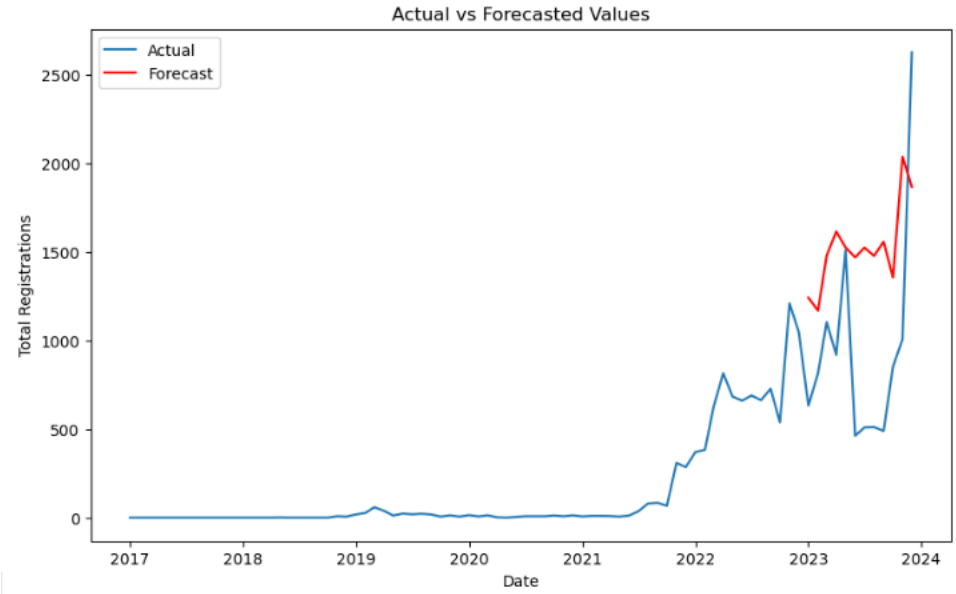


Figure 12: Line Graph on Actual and Forecasted Values – SARIMA

```
Mean Squared Error (MSE): 49870.71622159454
Root Mean Squared Error (RMSE): 223.3175233195876
R-squared (R2): 0.744883441408901
```

Figure 13: Performance Metrics - SARIMA

Figure above displays the predicted values as well as the actual values from the dataset. These forecasted values are from 2023-01-01 to 2023-12-01. Subsequently, we have compared the predicted and actual values, and we have found that the forecasted values are more realistic. It is giving an RMSE of 223.32 for the state of West Bengal.

5.3 Machine Learning Models

1. XGBOOST

	Date	Predicted_Value
0	2024-01-01	633.006531
1	2024-02-01	835.237671
2	2024-03-01	1103.978271
3	2024-04-01	919.030273
4	2024-05-01	1508.989014
5	2024-06-01	462.992218
6	2024-07-01	510.018097
7	2024-08-01	511.989502
8	2024-09-01	490.005432
9	2024-10-01	779.388367
10	2024-11-01	1008.006226
11	2024-12-01	2625.986816

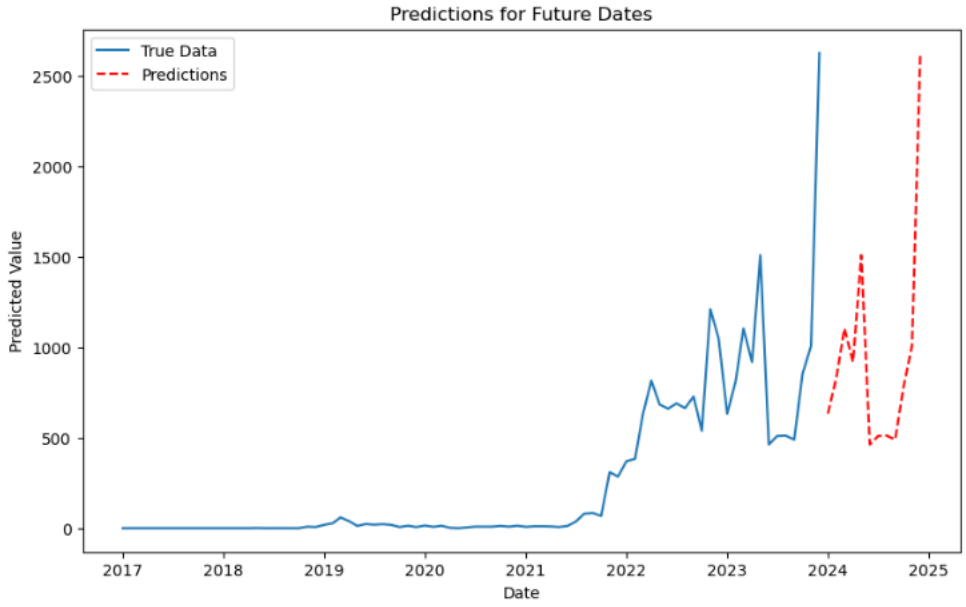


Figure 14: Line Graph on Predictions for Future Values – XGBOOST

```
Test RMSE: 144.65095874797765
Train R-squared (R2): 0.999999998142201
Test R-squared (R2): 0.8548223168045856
```

Figure 15: Performance Metrics – XGBOOST

Figure above displays the predicted values as well as the actual values from the dataset. The forecasted values are from 2024-01-01 to 2024-12-01. Subsequently, we have compared the predicted and actual values, and we have found that the

forecasted values are more realistic. It is giving an RMSE of 144.65 and R^2 is 85.48% for the state of West Bengal.

2. Random Forest

	Date	Predicted_Value
0	2024-01-01	632.005310
1	2024-02-01	719.638428
2	2024-03-01	1102.981567
3	2024-04-01	918.032043
4	2024-05-01	1507.974487
5	2024-06-01	462.003540
6	2024-07-01	509.022522
7	2024-08-01	510.952942
8	2024-09-01	489.031555
9	2024-10-01	643.111694
10	2024-11-01	1006.999512
11	2024-12-01	2624.999756

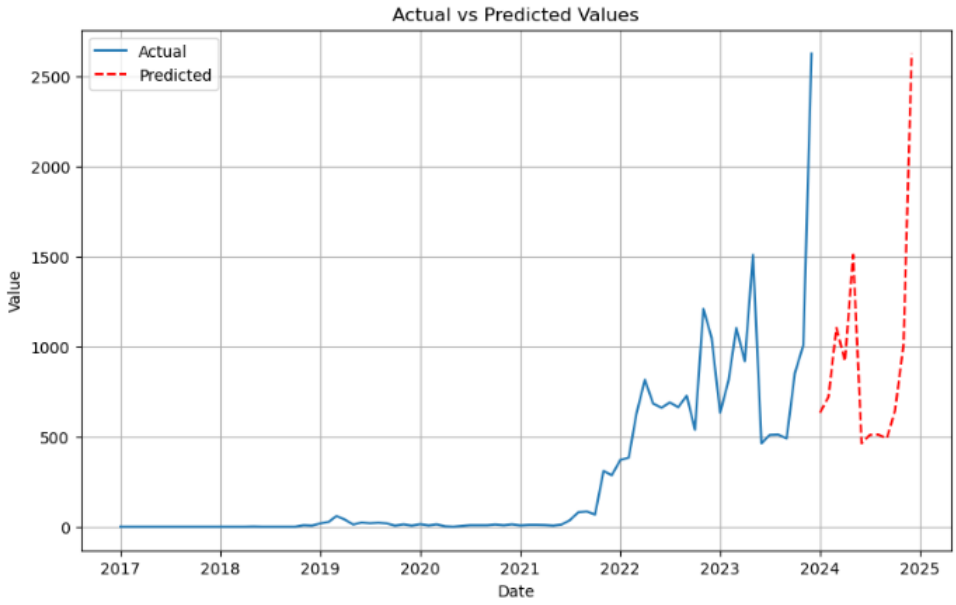


Figure 16: Line Graph on Predictions for Future Values – Random Forest

```
Test MAE: 92.5034221474025
Test MSE: 22932.060287456807
Test RMSE: 151.4333526256908
Test R2: 0.8408889640727304
```

Figure 17: Performance Metrics – Random Forest

Figure above displays the predicted values as well as the actual values from the dataset. The forecasted values are from 2024-01-01 to 2024-12-01. Subsequently, we have compared the predicted and actual values, and we have found that the forecasted values are more realistic. It is giving an RMSE of 151.43. and R^2 is 84.08% for the state of West Bengal.

CHAPTER 6: CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

As per the results, we noted that in time series model SARIMA is the best model as it provided with better performance (RMSE) in forecasting i.e., 223.32 compared to ARIMA. Further, it shows that the dataset is a seasonal data and thus SARIMA is the best model.

In Machine learning model, we noted that XGBOOST is the best fitting model as it provides better performance in R^2 i.e., 85.48% whereas random forest provides 84.08%.

6.2 Future Scope

As vehicles are very important for every individual and resources are depleting, Electric Vehicles would play a very crucial role in the future. There are various factors that would increase the sales of EVs in the future such as Government policies and initiatives, Environmental regulations, Cost reduction etc.

We can note that the sales forecasting for 2-Wheeler EVs are increasing for the future period. We can use this model for the purpose of forecasting for other EVs as well such as 3-Wheelers, 4-Wheelers etc. This would help in analyzing the demand and the requirement to manufacture the required number of EVs.

Further, we can expand this project by using other models such as Neural networks and other machine learning models to get more accuracy to the model.

REFERENCES

1. (n.d.). Retrieved from Timesofindia: https://timesofindia.indiatimes.com/travel/travel-news/report-india-ranks-3rd-in-worst-air-quality-delhi-remains-worlds-most-polluted-capital/articleshow/108608620.cms#:~:text=According%20to%20the%20World%20Air,concentration%2C%20following%20Bangladesh
2. (n.d.). Retrieved from thehindubusinessline: https://www.thehindubusinessline.com/companies/ev-sales-grow-by-48-in-cy2023/article67692457.ece
3. (n.d.). Retrieved from business-standard: https://www.business-standard.com/about/what-is-electric-vehicle
4. (n.d.). Retrieved from intellipaat: https://intellipaat.com/blog/advantages-and-disadvantages-of-electric-vehicles/
5. (n.d.). Retrieved from javatpoint: https://www.javatpoint.com/machine-learning-models
6. (n.d.). Retrieved from geeksforgeeks, ml: https://www.geeksforgeeks.org/ml-machine-learning/
7. (n.d.). Retrieved from tableau, time series: https://www.tableau.com/learn/articles/time-series-analysis
8. (n.d.). Retrieved from geeksforgeeks, SARIMA: https://www.geeksforgeeks.org/sarima-seasonal-autoregressive-integrated-moving-average/
9. (n.d.). Retrieved from geeksforgeeks, xgboost: https://www.geeksforgeeks.org/xgboost/
10. (n.d.). Retrieved from python: https://www.python.org/doc/essays/blurb/

11. (n.d.). Retrieved from monashdatafluency, powerbi: https://monashdatafluency.github.io/Power_BI/introduction-to-power-bi.html

12. (n.d.). Retrieved from sciencedirect, arima: https://www.sciencedirect.com/topics/mathematics/autoregressive-integrated-moving-average

13. (n.d.). Retrieved from realpython, jupyter: https://realpython.com/jupyter-notebook-introduction/

14. (n.d.). Retrieved from domino, jupyter: https://domino.ai/data-science-dictionary/jupyter-notebook

15. (n.d.). Retrieved from javatpoint, random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm

16. Ashok Jhunjhunwala, P. K. (2018). Electric Vehicles in India. *IEEE Electrification Magazine*.

17. Deep Karan Singh, N. R. (2023). Machine Learning for Weather Forecasting: XGBoost vs SVM vs Random Forest in Predicting Temperature for Visakhapatnam. *MECS Press*.

18. Du, Z. Z. (2018). Forecasting the Number of Electric Vehicles: A Case of Beijing. *IOP Publishing*.

19. Himdi, H. O. (2023). Comparing Machine Learning Methods—SVR, XGBoost, LSTM, and MLP— For Forecasting the Moroccan Stock Market. *Computer science and mathematics forum*.

20. Iqbal Husain, M. S. (2020). Electric Drive Technology Trends, Challenges, and Opportunities for Future Electric Vehicles. *Invited Paper*.

21. Julian Hubera, D. D. (2020). Probabilistic forecasts of time and energy flexibility in battery electric vehicle charging. *Applied energy*.

22. Julio A. Sanguesa 1, V. T.-S. (2021). A Review on Electric Vehicles: Technologies and Challenges. *Smart Cities*.

23. Khulood Albeladi, B. Z. (2023). Time Series Forecasting using LSTM and ARIMA. *International Journal of Advanced Computer Science and Applications*.

24. Kim, Y. K. (2021). Forecasting Charging Demand of Electric Vehicles Using Time-Series Models. *energies*.

25. Meiqin MAO, S. Z. (2019). Schedulable capacity forecasting for electric vehicles based on big data analysis. *J. Mod. Power Syst. Clean Energy*.

26. Ning Wang, L. T. (2018). A global comparison and assessment of incentive policy on electric vehicle promotion. *Sustainable Cities and Society*.

27. Nur Liyana Mohd Jailani, J. K.-S. (2023). Investigating the Power of LSTM-Based Models in Solar. *processes*.

28. Phumudzo Lloyd Seabe, C. R. (2023). Forecasting Cryptocurrency Prices Using LSTM, GRU, and Bi-Directional LSTM: A Deep Learning Approach. *fractal and fractional*.

29. Rajeev Ranjan Kumar, K. A. (2019). Adoption of Electric Vehicles: A Literature Review and Prospects for Sustainability. *Cleaner Production*.

30. *Regression*. (n.d.). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/regression-in-machine-learning/

31. *timesofindia*. (n.d.). Retrieved from https://timesofindia.indiatimes.com/auto/cars/evs-get-big-price-cuts-prices-running-cost-of-petrol-vs-electric cars/articleshow/107687660.cms

32. Xiaodong Yuan, Y. C. (2021). Forecasting the development trend of low emission vehicle technologies: Based on patent data. *Technological Forecasting & Social Change*.

33. Xiaoli Sun, Z. L. (2019). Technology Development of Electric Vehicles: A Review. *energies*.