Electric Vehicle Sales by State in India

This project aims to analyze and predict the sales of Electric Vehicles (EV) by state in India using machine learning.

Dataset Information

This dataset is valuable for analysts, data scientists, and researchers aiming to understand electric vehicle (EV) adoption trends across India. It is versatile and ideal for geographic market segmentation, trend analysis, and predictive modeling. By offering insights into regional EV sales patterns, the dataset supports strategic decision-making in market planning and infrastructure investment. The data was meticulously scraped from the Clean Mobility Shift website, and then thoroughly preprocessed to ensure accuracy and relevance. All null values have been removed, and the dataset has been cleaned to prepare it for immediate use in exploration, visualization, and analytical projects. It is particularly valuable for market trend analysis, infrastructure planning, and policy development within the EV sector.

The dataset contains the following columns:

- Year: The year of the sales.
- Month_Name: The month in which sales occurred.
- Date: The specific date of the sales.
- State: The state in India where the sales occurred.
- Vehicle_Class: The class of the vehicle (e.g., sedan, SUV, etc.).
- Vehicle_Category: The category of the vehicle (e.g., commercial, passenger).
- Vehicle_Type: The type of the vehicle (e.g., 2-wheeler, 4-wheeler).
- EV_Sales_Quantity: The quantity of EV sales.

Data loading

Load the "Electric Vehicle Sales by State in India.csv" file into a pandas DataFrame.

```
import pandas as pd

df = pd.read_csv('Electric Vehicle Sales by State in India.csv')
display(df.head())
```

```
      Year Month_Name      Date           State         Vehicle_Class  \
0  2014.0        jan  1/1/2014  Andhra Pradesh       ADAPTED VEHICLE
1  2014.0        jan  1/1/2014  Andhra Pradesh  AGRICULTURAL TRACTOR
2  2014.0        jan  1/1/2014  Andhra Pradesh             AMBULANCE
3  2014.0        jan  1/1/2014  Andhra Pradesh   ARTICULATED VEHICLE
4  2014.0        jan  1/1/2014  Andhra Pradesh                   BUS

  Vehicle_Category Vehicle_Type  EV_Sales_Quantity
0           Others       Others                0.0
1           Others       Others                0.0
2           Others       Others                0.0
3           Others       Others                0.0
4              Bus          Bus                0.0
```

Data exploration

Explore the data by examining its shape, data types, missing values, descriptive statistics, and unique values in categorical columns.

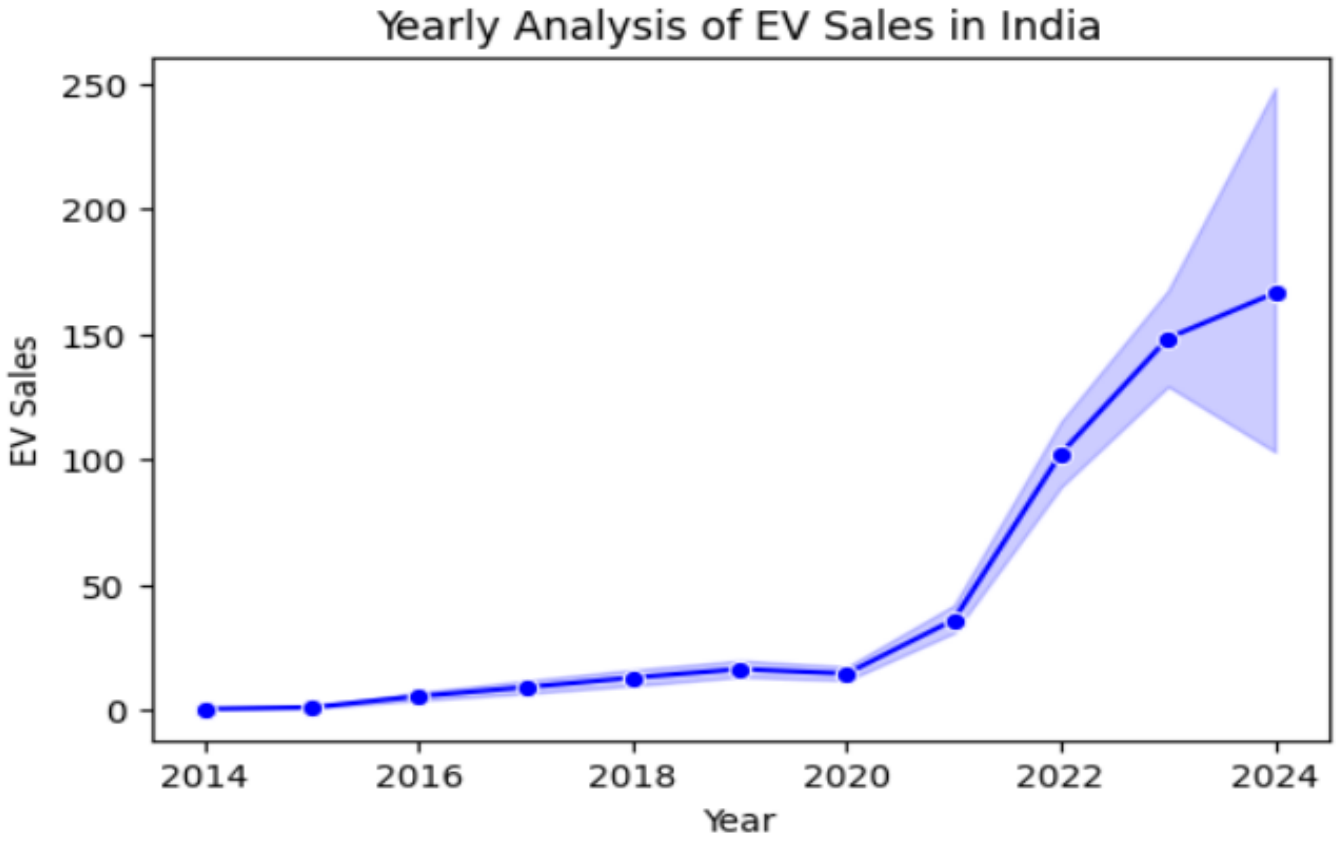
```
# Examine the DataFrame's shape
print("DataFrame Shape:", df.shape)

# View the data types of each column
print("\nData Types:")
print(df.dtypes)

# Check for missing values
print("\nMissing Values:")
print(df.isnull().sum())

# Explore the descriptive statistics
print("\nDescriptive Statistics:")
print(df.describe())

# Analyze the unique values for categorical columns
print("\nUnique States:", df['State'].unique())
print("\nUnique Vehicle Classes:", df['Vehicle_Class'].unique())
print("\nUnique Vehicle Categories:", df['Vehicle_Category'].unique())
```

```
DataFrame Shape: (96845, 8)

Data Types:
Year                 float64
Month_Name            object
Date                  object
State                 object
Vehicle_Class         object
Vehicle_Category      object
Vehicle_Type          object
EV_Sales_Quantity    float64
dtype: object

Missing Values:
Year                 0
Month_Name           0
Date                 0
State                0
Vehicle_Class        0
Vehicle_Category     0
Vehicle_Type         0
EV_Sales_Quantity    0
dtype: int64

Descriptive Statistics:
                Year  EV_Sales_Quantity
count  96845.000000       96845.000000

mean    2018.622768          37.108896
std        2.895581         431.566675
min     2014.000000           0.000000
25%     2016.000000           0.000000
50%     2019.000000           0.000000
75%     2021.000000           0.000000
max     2024.000000       20584.000000

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

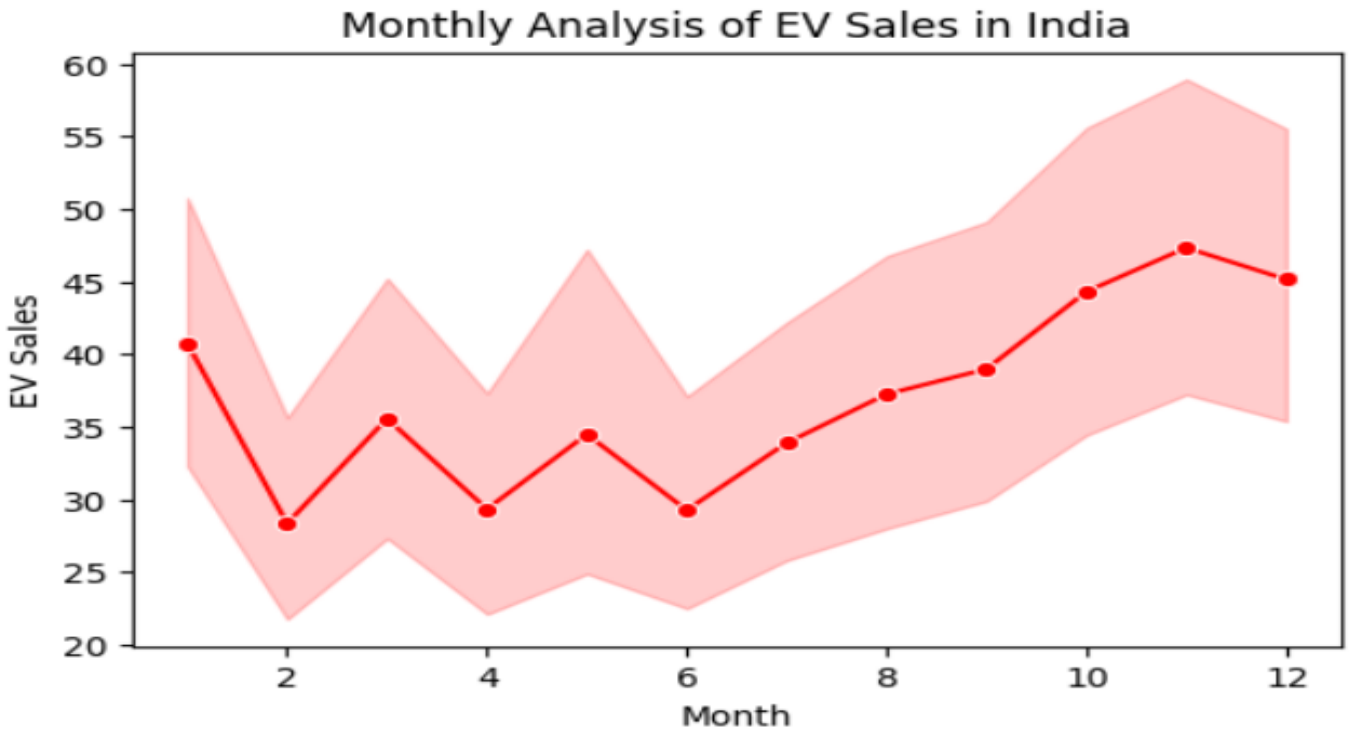
```
Unique Vehicle Classes: ['ADAPTED VEHICLE' 'AGRICULTURAL TRACTOR' 'AMBULANCE'
 'ARTICULATED VEHICLE' 'BUS' 'CASH VAN' 'CRANE MOUNTED VEHICLE'
 'EDUCATIONAL INSTITUTION BUS' 'EXCAVATOR (COMMERCIAL)' 'FORK LIFT'
 'GOODS CARRIER' 'HARVESTER' 'MAXI CAB' 'M-CYCLE/SCOOTER' 'MOTOR CAB'
 'MOTOR CAR' 'OMNI BUS' 'PRIVATE SERVICE VEHICLE' 'RECOVERY VEHICLE'
 'THREE WHEELER (PASSENGER)' 'TRACTOR (COMMERCIAL)'
 'TRAILER (AGRICULTURAL)' 'TRAILER (COMMERCIAL)'
 'TRAILER FOR PERSONAL USE' 'VEHICLE FITTED WITH RIG'
 'CAMPER VAN / TRAILER' 'CONSTRUCTION EQUIPMENT VEHICLE (COMMERCIAL)'
 'DUMPER' 'EXCAVATOR (NT)' 'MOPED' 'THREE WHEELER (PERSONAL)'
 'THREE WHEELER (GOODS)' 'EARTH MOVING EQUIPMENT'
 'MOTOR CYCLE/SCOOTER-USED FOR HIRE' 'CONSTRUCTION EQUIPMENT VEHICLE'
 'M-CYCLE/SCOOTER-WITH SIDE CAR' 'MOBILE WORKSHOP'
 'OMNI BUS (PRIVATE USE)' 'VEHICLE FITTED WITH COMPRESSOR'
 'CAMPER VAN / TRAILER (PRIVATE USE)' 'LUXURY CAB'
 'MOTOR CYCLE/SCOOTER-SIDECAR(T)' 'ANIMAL AMBULANCE' 'BREAKDOWN VAN'
 'FIRE FIGHTING VEHICLE' 'TOW TRUCK' 'TRACTOR-TROLLEY(COMMERCIAL)'
 'PRIVATE SERVICE VEHICLE (INDIVIDUAL USE)' 'BULLDOZER' 'HEARSES'
 'MOBILE CLINIC' 'MOTORISED CYCLE (CC > 25CC)' 'POWER TILLER'
 'POWER TILLER (COMMERCIAL)' 'VEHICLE FITTED WITH GENERATOR' 'LIBRARY VAN'
 'ROAD ROLLER' 'E-RICKSHAW(P)' 'FIRE TENDERS' 'TOWER WAGON'
 'TREE TRIMMING VEHICLE' 'MOBILE CANTEEN' 'AUXILIARY TRAILER'
 'E-RICKSHAW WITH CART (G)' 'X-RAY VAN' 'SNORKED LADDERS'
 'QUADRICYCLE (COMMERCIAL)' 'QUADRICYCLE (PRIVATE)'
 'SEMI-TRAILER (COMMERCIAL)' 'ARMOURED/SPECIALISED VEHICLE'
 'MOTOR CYCLE/SCOOTER-WITH TRAILER' 'MODULAR HYDRAULIC TRAILER'
 'MOTOR CARAVAN']

Unique Vehicle Categories: ['Others' 'Bus' '2-Wheelers' '4-Wheelers'
'3-Wheelers']
```

Based on the initial exploration, create histograms for numerical features like “EV_Sales_Quantity” to visualize their distribution.

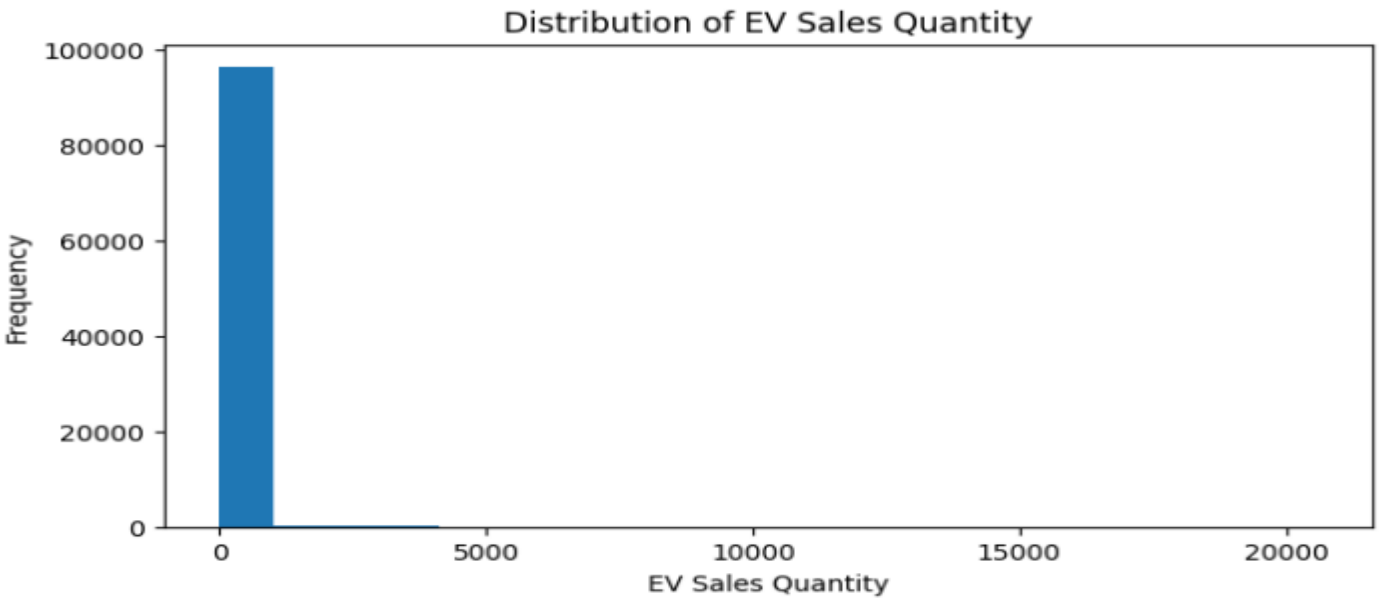
```python
#Data Visualisation
import seaborn as sns
plt.figure(figsize=(6,4))
plt.title('Yearly Analysis of EV Sales in India')
sns.lineplot(x='Year', y='EV_Sales_Quantity', data=df,marker='o', color='b')
plt.xlabel('Year')
plt.ylabel('EV Sales');
```




```
plt.figure(figsize=(6,4))
plt.title('Monthly Analysis of EV Sales in India')
sns.lineplot(x='Month_Name', y='EV_Sales_Quantity', data=df,marker='o',␣
 ↪color='r')
plt.xlabel('Month')
plt.ylabel('EV Sales');
```



```
import matplotlib.pyplot as plt

# Visualize the distribution of EV_Sales_Quantity
plt.figure(figsize=(8, 4))
plt.hist(df['EV_Sales_Quantity'], bins=20)
plt.xlabel('EV Sales Quantity')
plt.ylabel('Frequency')
plt.title('Distribution of EV Sales Quantity')
plt.show()
```



Data cleaning

Remove duplicate rows, convert ‘Year’ and ‘Month_Name’ to datetime format, convert ‘EV_Sales_Quantity’ to integer type, and handle inconsistencies in categorical columns.

```
# Remove duplicate rows
df.drop_duplicates(inplace=True)

# Convert 'Year' and 'Month_Name' to datetime format
df['Year'] = pd.to_datetime(df['Year'], format='%Y', errors='coerce').dt.year
df['Month_Name'] = pd.to_datetime(df['Month_Name'], format='%b',␣
 ↪errors='coerce').dt.month

# Convert 'EV_Sales_Quantity' to integer type
df['EV_Sales_Quantity'] = df['EV_Sales_Quantity'].astype(int)

# Handle potential inconsistencies in State, Vehicle_Class, and Vehicle_Category
df['State'] = df['State'].str.strip().str.title()
df['Vehicle_Class'] = df['Vehicle_Class'].str.strip().str.title()
df['Vehicle_Category'] = df['Vehicle_Category'].str.strip().str.title()
```

```python
# Check for missing values after cleaning
print("Missing Values after cleaning:")
print(df.isnull().sum())

# Check unique values for categorical columns after cleaning
print("\nUnique States:", df['State'].unique())
print("\nUnique Vehicle Classes:", df['Vehicle_Class'].unique())
print("\nUnique Vehicle Categories:", df['Vehicle_Category'].unique())

# Check data types after cleaning
print("\nData Types after cleaning:")
print(df.dtypes)

# Display the first few rows of the cleaned DataFrame
display(df.head())
```

```
Missing Values after cleaning:
Year                0
Month_Name          0
Date                0
State               0
Vehicle_Class       0
Vehicle_Category    0
Vehicle_Type        0
EV_Sales_Quantity   0
dtype: int64

Unique States: ['Andhra Pradesh' 'Arunachal Pradesh' 'Assam' 'Andaman & Nicobar
Island'
 'Bihar' 'Chhattisgarh' 'Chandigarh' 'Dnh And Dd' 'Delhi' 'Goa' 'Gujarat'
 'Himachal Pradesh' 'Haryana' 'Jharkhand' 'Karnataka' 'Kerala' 'Ladakh'
 'Maharashtra' 'Meghalaya' 'Madhya Pradesh' 'Mizoram' 'Nagaland' 'Odisha'
 'Punjab' 'Puducherry' 'Rajasthan' 'Sikkim' 'Tamil Nadu' 'Tripura'
 'Uttarakhand' 'Uttar Pradesh' 'West Bengal' 'Jammu And Kashmir' 'Manipur']


Unique Vehicle Classes: ['Adapted Vehicle' 'Agricultural Tractor' 'Ambulance'
 'Articulated Vehicle' 'Bus' 'Cash Van' 'Crane Mounted Vehicle'
 'Educational Institution Bus' 'Excavator (Commercial)' 'Fork Lift'
 'Goods Carrier' 'Harvester' 'Maxi Cab' 'M-Cycle/Scooter' 'Motor Cab'
 'Motor Car' 'Omni Bus' 'Private Service Vehicle' 'Recovery Vehicle'
 'Three Wheeler (Passenger)' 'Tractor (Commercial)'
 'Trailer (Agricultural)' 'Trailer (Commercial)'
 'Trailer For Personal Use' 'Vehicle Fitted With Rig'
 'Camper Van / Trailer' 'Construction Equipment Vehicle (Commercial)'
 'Dumper' 'Excavator (Nt)' 'Moped' 'Three Wheeler (Personal)'
 'Three Wheeler (Goods)' 'Earth Moving Equipment'
 'Motor Cycle/Scooter-Used For Hire' 'Construction Equipment Vehicle'
```

```
 'M-Cycle/Scooter-With Side Car' 'Mobile Workshop'
 'Omni Bus (Private Use)' 'Vehicle Fitted With Compressor'
 'Camper Van / Trailer (Private Use)' 'Luxury Cab'
 'Motor Cycle/Scooter-Sidecar(T)' 'Animal Ambulance' 'Breakdown Van'
 'Fire Fighting Vehicle' 'Tow Truck' 'Tractor-Trolley(Commercial)'
 'Private Service Vehicle (Individual Use)' 'Bulldozer' 'Hearses'
 'Mobile Clinic' 'Motorised Cycle (Cc > 25Cc)' 'Power Tiller'
 'Power Tiller (Commercial)' 'Vehicle Fitted With Generator' 'Library Van'
 'Road Roller' 'E-Rickshaw(P)' 'Fire Tenders' 'Tower Wagon'
 'Tree Trimming Vehicle' 'Mobile Canteen' 'Auxiliary Trailer'
 'E-Rickshaw With Cart (G)' 'X-Ray Van' 'Snorked Ladders'
 'Quadricycle (Commercial)' 'Quadricycle (Private)'
 'Semi-Trailer (Commercial)' 'Armoured/Specialised Vehicle'
 'Motor Cycle/Scooter-With Trailer' 'Modular Hydraulic Trailer'
 'Motor Caravan']

Unique Vehicle Categories: ['Others' 'Bus' '2-Wheelers' '4-Wheelers'
'3-Wheelers']

Data Types after cleaning:
Year                  int32
Month_Name            int32
Date                 object
State                object
Vehicle_Class        object
Vehicle_Category     object
Vehicle_Type         object
EV_Sales_Quantity     int64
dtype: object
```

	Year	Month_Name	Date	State	Vehicle_Class
0	2014	1	1/1/2014	Andhra Pradesh	Adapted Vehicle
1	2014	1	1/1/2014	Andhra Pradesh	Agricultural Tractor
2	2014	1	1/1/2014	Andhra Pradesh	Ambulance
3	2014	1	1/1/2014	Andhra Pradesh	Articulated Vehicle
4	2014	1	1/1/2014	Andhra Pradesh	Bus

	Vehicle_Category	Vehicle_Type	EV_Sales_Quantity
0	Others	Others	0
1	Others	Others	0
2	Others	Others	0
3	Others	Others	0
4	Bus	Bus	0

Data wrangling

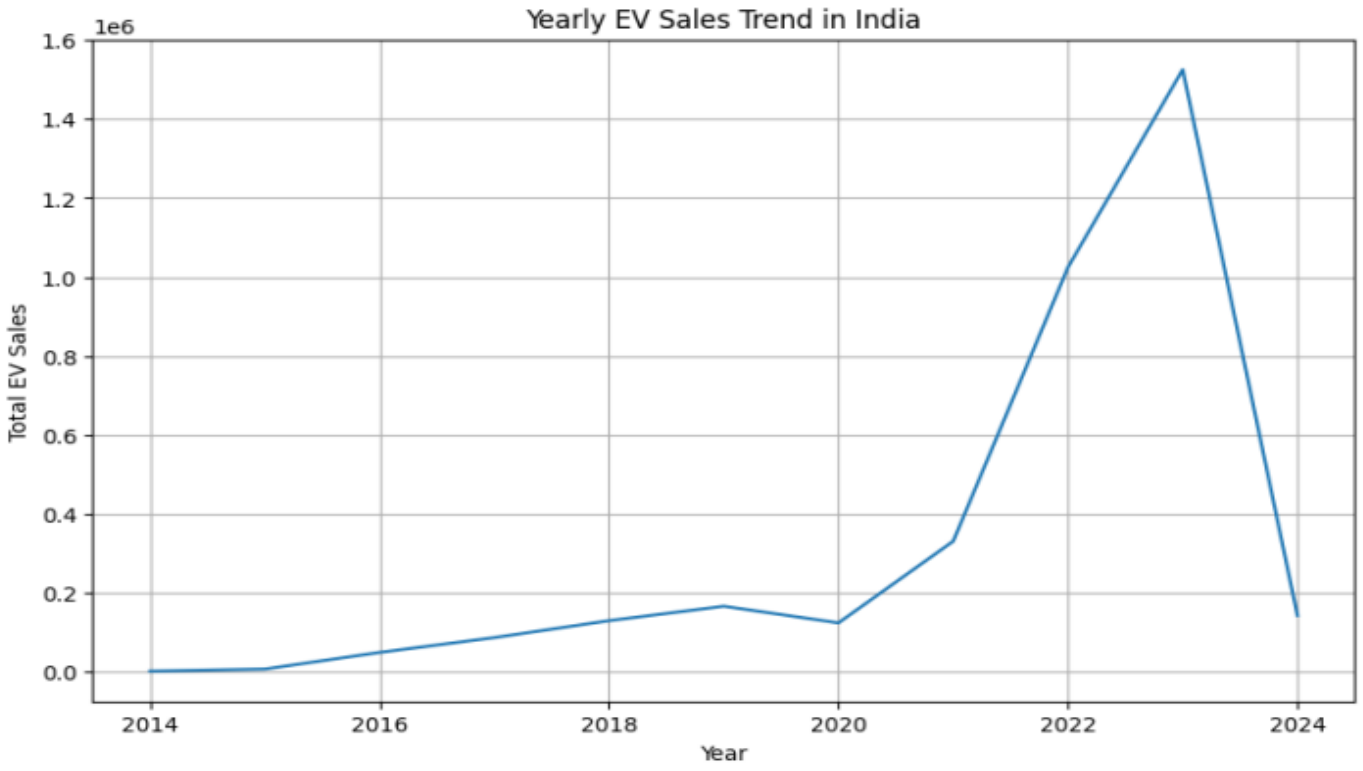
Combine ‘Year’ and ‘Month_Name’ into a new ‘Date’ column, then extract ‘Year’ and ‘Month’ from the ‘Date’ column to prepare the data for analysis.

```
# Combine 'Year' and 'Month_Name' into a new 'Date' column
df['Date'] = pd.to_datetime(df['Year'].astype(str) + '-' + df['Month_Name'].
 ↪astype(str) + '-01')

# Extract 'Year' and 'Month' from 'Date'
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
```

Data analysis

Perform yearly analysis by grouping the data by ‘Year’ and calculating the total EV sales for each year. Then, plot the total EV sales over time to observe the overall yearly growth trend.

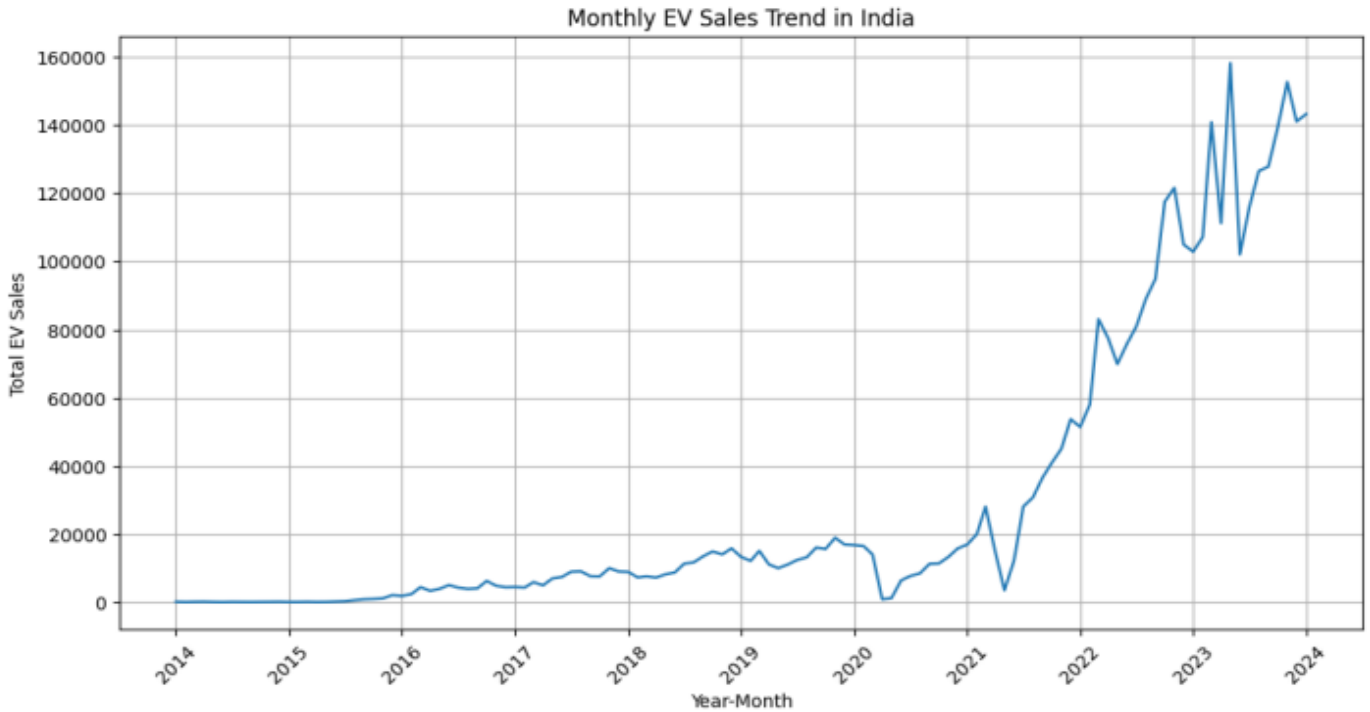
```
# Yearly Analysis
yearly_sales = df.groupby('Year')['EV_Sales_Quantity'].sum()

# Plot the total EV sales over time
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.plot(yearly_sales.index, yearly_sales.values)
plt.xlabel('Year')
plt.ylabel('Total EV Sales')
plt.title('Yearly EV Sales Trend in India')
plt.grid(True)
plt.show()
```



Perform monthly analysis by grouping the data by ‘Year’ and ‘Month’ and calculate the total EV sales for each month. Then plot the monthly EV sales over time to observe seasonal patterns and monthly fluctuations.

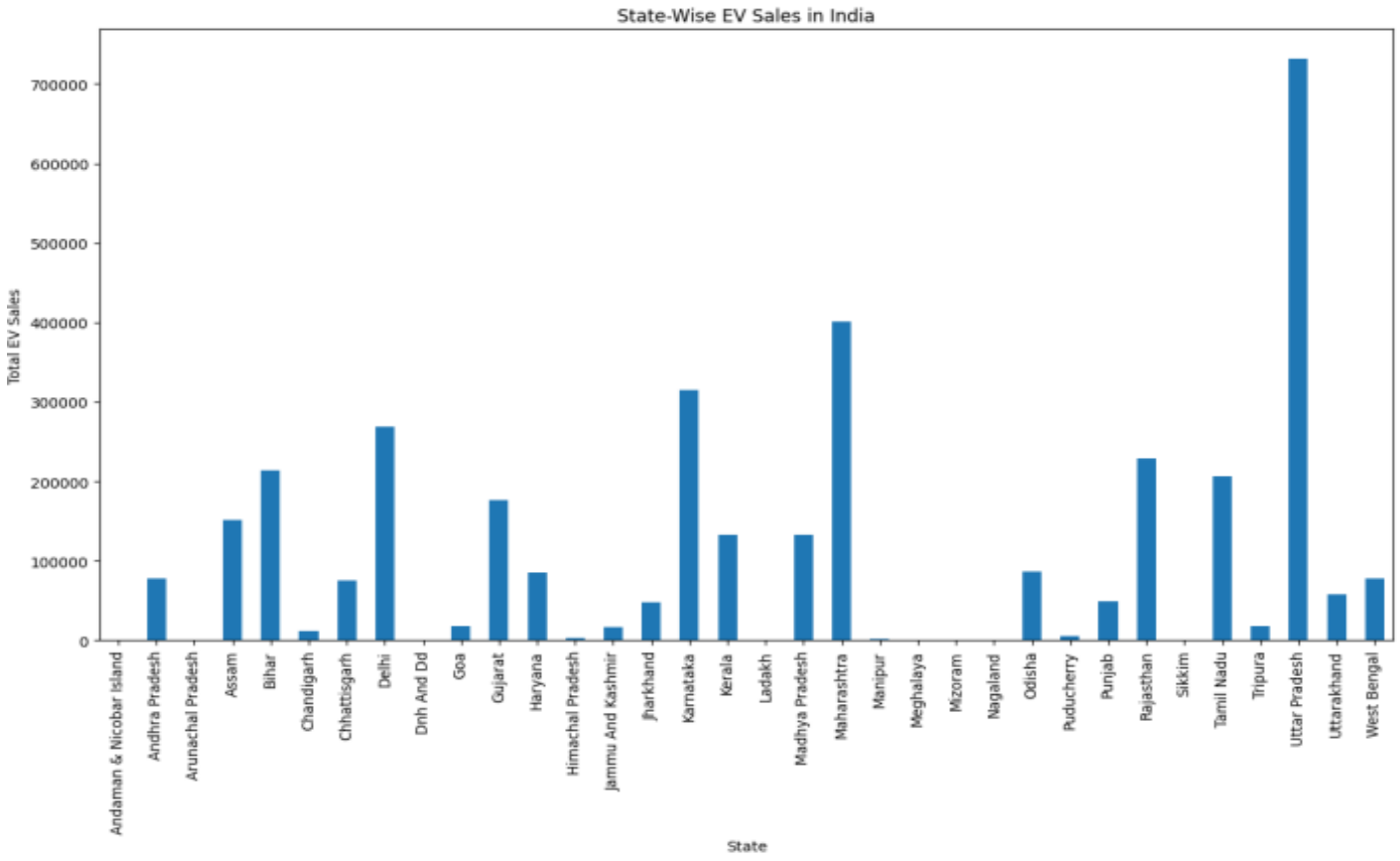
```
# Monthly Analysis
monthly_sales = df.groupby(['Year', 'Month'])['EV_Sales_Quantity'].sum()

# Reset the index to convert the MultiIndex to columns
monthly_sales = monthly_sales.reset_index()

# Create a datetime column for plotting
monthly_sales['Year-Month'] = pd.to_datetime(monthly_sales['Year'].astype(str)␣
  ↪+ '-' + monthly_sales['Month'].astype(str) + '-01')


# Plot the monthly EV sales over time
plt.figure(figsize=(12, 6))
# Plot using the new 'Year-Month' column for x-axis
plt.plot(monthly_sales['Year-Month'], monthly_sales['EV_Sales_Quantity'])
plt.xlabel('Year-Month')
plt.ylabel('Total EV Sales')
plt.title('Monthly EV Sales Trend in India')
plt.grid(True)
plt.xticks(rotation=45)
```

```
plt.show()
```



Perform state-wise analysis by grouping the data by ‘State’ and calculating the total EV sales for each state. Then identify the states with the highest and lowest EV sales and plot a bar chart to visualize the state-wise EV sales.

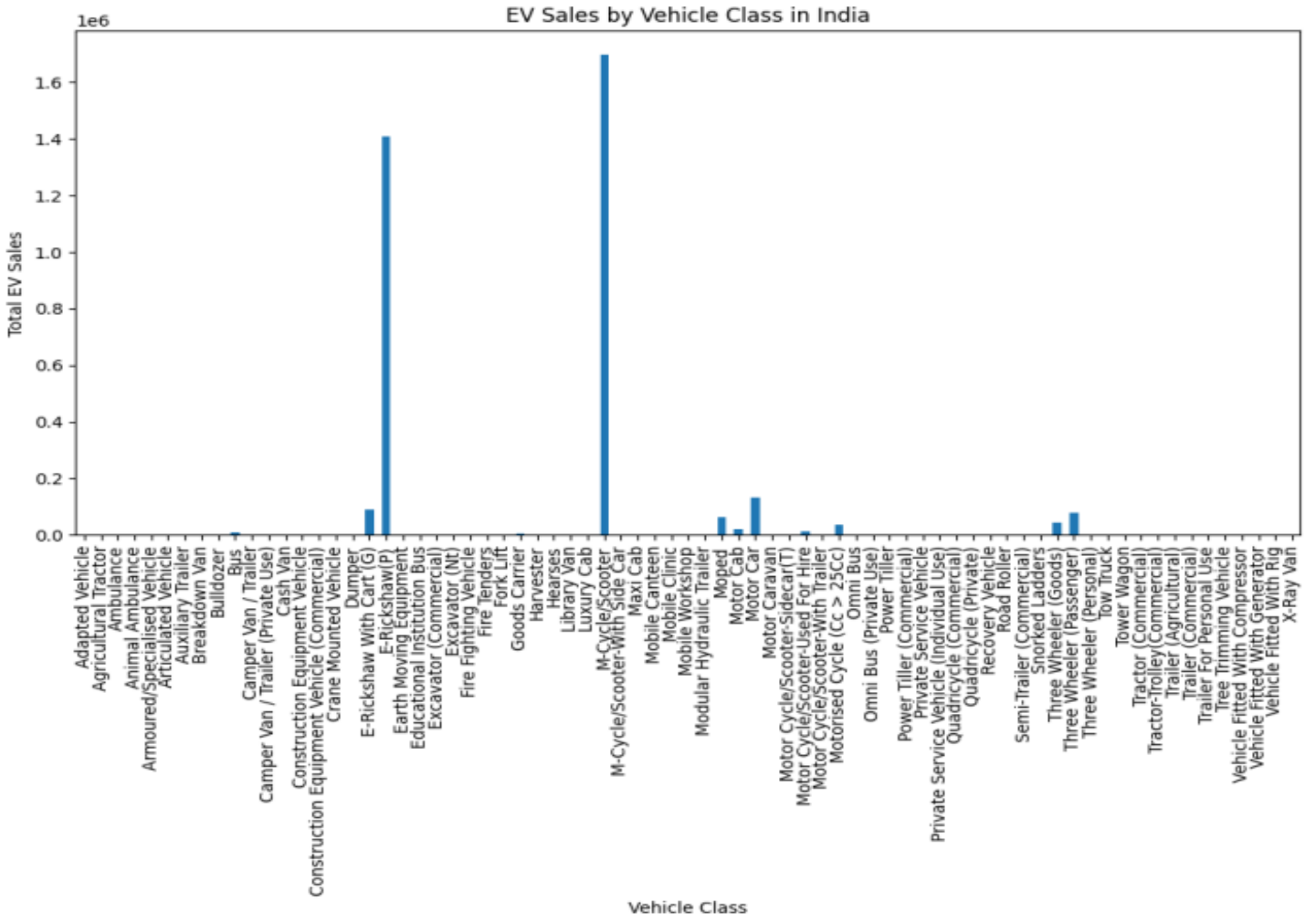
```
# State-Wise Analysis
state_wise_sales = df.groupby('State')['EV_Sales_Quantity'].sum()

# Identify the states with the highest and lowest EV sales
highest_sales_state = state_wise_sales.idxmax()
lowest_sales_state = state_wise_sales.idxmin()

# Plot a bar chart to visualize the state-wise EV sales
plt.figure(figsize=(15, 8))
state_wise_sales.plot(kind='bar')
plt.xlabel('State')
plt.ylabel('Total EV Sales')
plt.title('State-Wise EV Sales in India')
plt.xticks(rotation=90)
plt.show()

print(f"State with highest EV sales: {highest_sales_state}")
print(f"State with lowest EV sales: {lowest_sales_state}")
```




```
State with highest EV sales: Uttar Pradesh
State with lowest EV sales: Sikkim
```

Perform vehicle class analysis by grouping the data by ‘Vehicle_Class’ and calculate the total EV sales for each class. Then identify the vehicle class with the highest sales and plot a bar chart to visualize EV sales by vehicle class.

```
# Vehicle Class Analysis
vehicle_class_sales = df.groupby('Vehicle_Class')['EV_Sales_Quantity'].sum()

# Identify the vehicle class with the highest sales
highest_sales_vehicle_class = vehicle_class_sales.idxmax()

# Plot a bar chart to visualize EV sales by vehicle class
plt.figure(figsize=(12, 6))
vehicle_class_sales.plot(kind='bar')
plt.xlabel('Vehicle Class')
plt.ylabel('Total EV Sales')
plt.title('EV Sales by Vehicle Class in India')
plt.xticks(rotation=90)
plt.show()

print(f"Vehicle class with highest EV sales: {highest_sales_vehicle_class}")
```

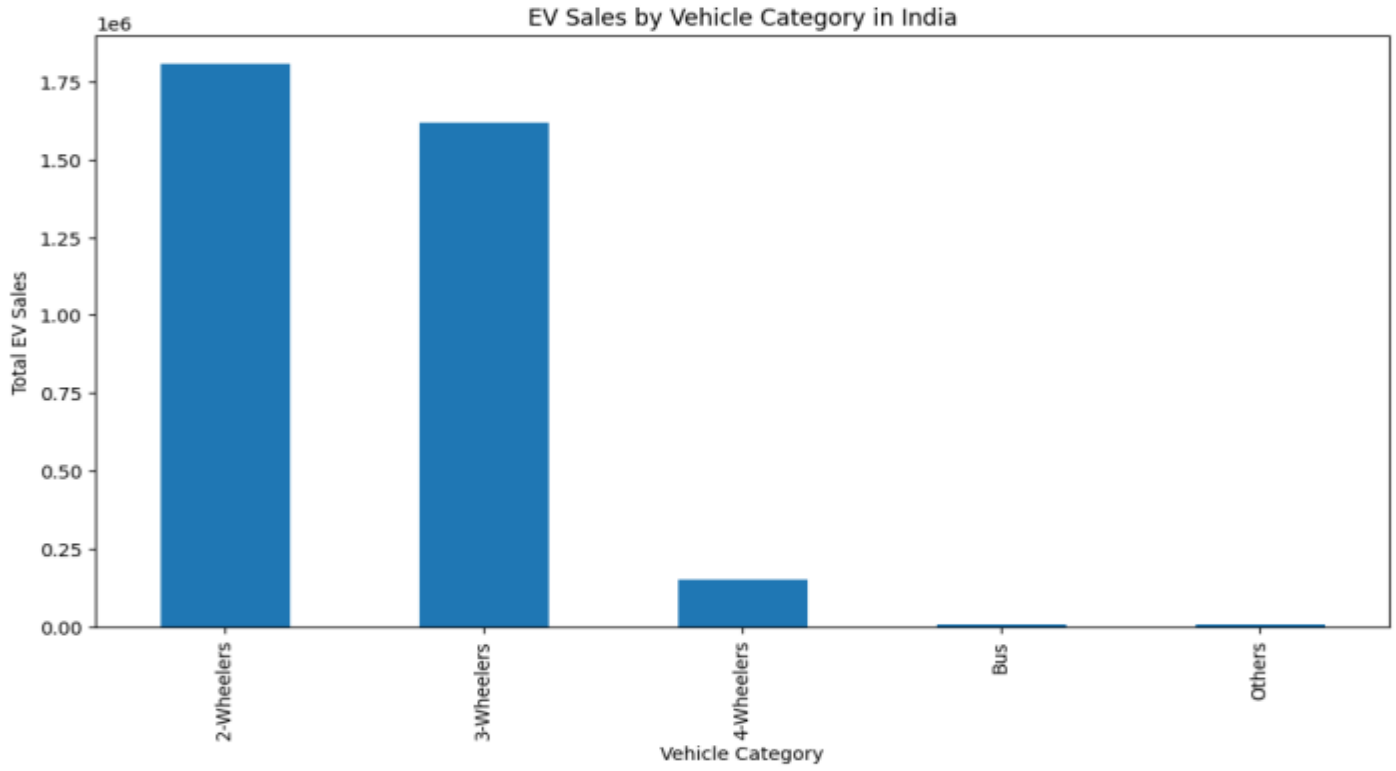



```
Vehicle class with highest EV sales: M-Cycle/Scooter
```

Perform vehicle category analysis by grouping the data by ‘Vehicle_Category’ and calculate the total EV sales for each category. Then identify the vehicle category with the highest sales and plot a bar chart to visualize EV sales by vehicle category.

```
# Vehicle Category Analysis
vehicle_category_sales = df.groupby('Vehicle_Category')['EV_Sales_Quantity'].
 ↪sum()

# Identify the vehicle category with the highest sales
highest_sales_vehicle_category = vehicle_category_sales.idxmax()

# Plot a bar chart to visualize EV sales by vehicle category
plt.figure(figsize=(12, 6))
vehicle_category_sales.plot(kind='bar')
plt.xlabel('Vehicle Category')
plt.ylabel('Total EV Sales')
```

```
plt.title('EV Sales by Vehicle Category in India')
plt.xticks(rotation=90)
plt.show()

print(f"Vehicle category with highest EV sales:␣
 ↪{highest_sales_vehicle_category}")
```



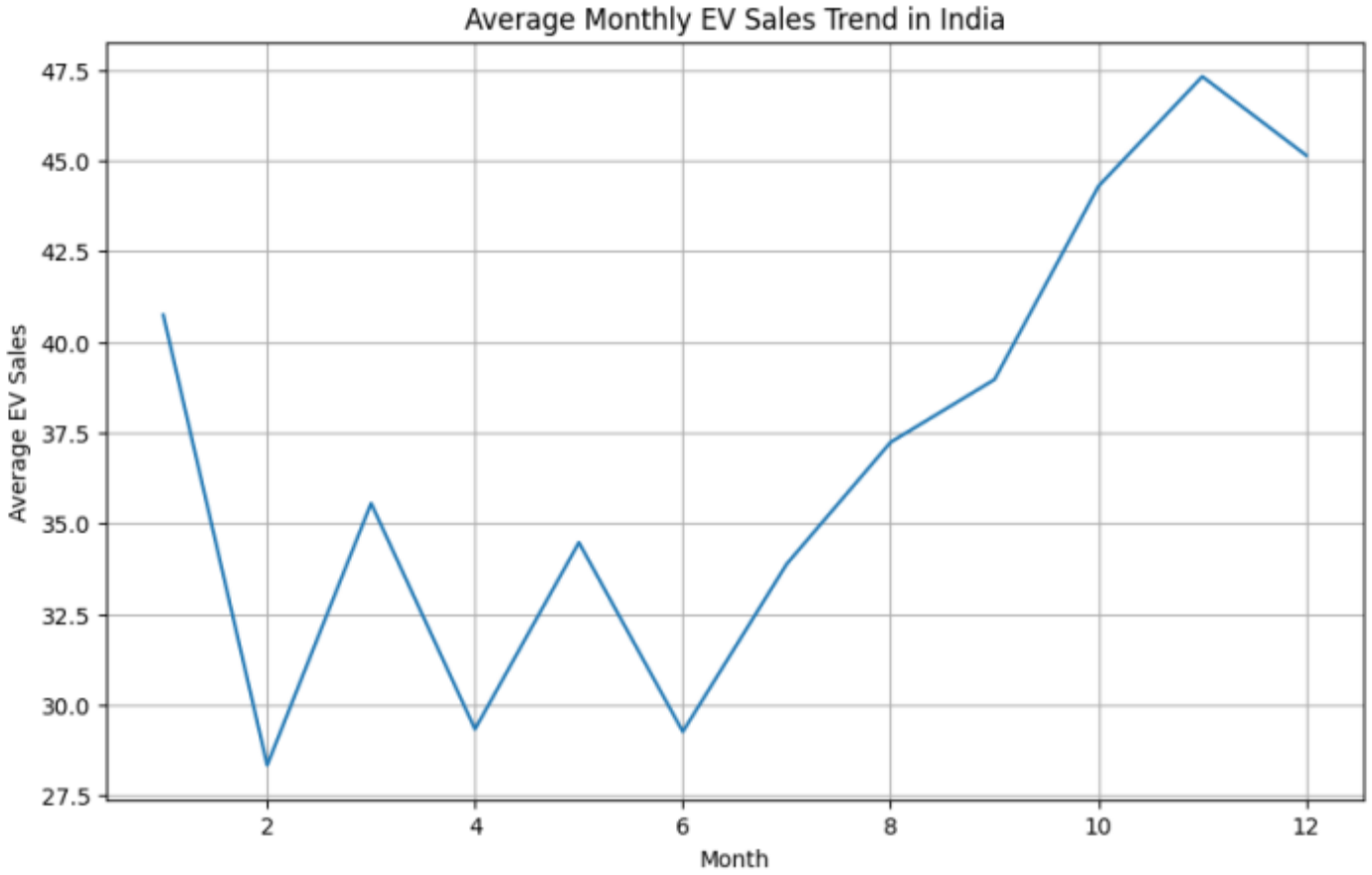

```
Vehicle category with highest EV sales: 2-Wheelers
```

Data visualization

Create visualizations for yearly, monthly, state-wise, vehicle class, and vehicle category EV sales trends in India.

```
# Monthly Analysis
monthly_sales = df.groupby('Month')['EV_Sales_Quantity'].mean()

# Plot the average monthly EV sales over the years
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.plot(monthly_sales.index, monthly_sales.values)
plt.xlabel('Month')
plt.ylabel('Average EV Sales')
```

```
plt.title('Average Monthly EV Sales Trend in India')
plt.grid(True)
plt.show()
```



Feature engineering

```
# Create lagged sales features
df['Lagged_Sales_1M'] = df.groupby(['State', 'Vehicle_Class',␣
 ↪'Vehicle_Category'])['EV_Sales_Quantity'].shift(1)
df['Lagged_Sales_3M'] = df.groupby(['State', 'Vehicle_Class',␣
 ↪'Vehicle_Category'])['EV_Sales_Quantity'].shift(3)
df['Lagged_Sales_6M'] = df.groupby(['State', 'Vehicle_Class',␣
 ↪'Vehicle_Category'])['EV_Sales_Quantity'].shift(6)
df['Lagged_Sales_12M'] = df.groupby(['State', 'Vehicle_Class',␣
 ↪'Vehicle_Category'])['EV_Sales_Quantity'].shift(12)

# One-hot encode categorical features
df = pd.get_dummies(df, columns=['State', 'Vehicle_Class', 'Vehicle_Category'],␣
 ↪prefix=['State', 'Vehicle_Class', 'Vehicle_Category'])

# Create time-based features
df['Quarter'] = df['Date'].dt.quarter
```

```
   EV_Sales_Quantity  Lagged_Sales_1M  Lagged_Sales_3M  Lagged_Sales_6M  \
0                  0              NaN              NaN              NaN
1                  0              NaN              NaN              NaN
2                  0              NaN              NaN              NaN
3                  0              NaN              NaN              NaN
4                  0              NaN              NaN              NaN

   Lagged_Sales_12M  State_Andaman & Nicobar Island  State_Andhra Pradesh  \
0               NaN                           False                  True
1               NaN                           False                  True
2               NaN                           False                  True
3               NaN                           False                  True
4               NaN                           False                  True

   State_Arunachal Pradesh  State_Assam  State_Bihar  …  \
0                    False        False        False  …
1                    False        False        False  …
2                    False        False        False  …
3                    False        False        False  …
4                    False        False        False  …

   Vehicle_Class_Vehicle Fitted With Rig  Vehicle_Class_X-Ray Van  \
0                                  False                    False
1                                  False                    False
2                                  False                    False
3                                  False                    False
4                                  False                    False

   Vehicle_Category_2-Wheelers  Vehicle_Category_3-Wheelers  \
0                        False                        False
1                        False                        False
2                        False                        False
3                        False                        False
4                        False                        False

   Vehicle_Category_4-Wheelers  Vehicle_Category_Bus  Vehicle_Category_Others  \
0                        False                 False                     True
1                        False                 False                     True
```

```
# Combine features into a new DataFrame
feature_df = df[['EV_Sales_Quantity', 'Lagged_Sales_1M', 'Lagged_Sales_3M',␣
 ↪'Lagged_Sales_6M', 'Lagged_Sales_12M'] + \
               [col for col in df.columns if 'State_' in col or␣
 ↪'Vehicle_Class_' in col or 'Vehicle_Category_' in col] + \
               ['Year', 'Month', 'Quarter']]

display(feature_df.head())
```

```
2                      False                False                     True
3                      False                False                     True
4                      False                 True                    False

   Year  Month  Quarter
0  2014      1        1
1  2014      1        1
2  2014      1        1
3  2014      1        1
4  2014      1        1

[5 rows x 120 columns]
```

Data splitting

Split the feature_df DataFrame into training and testing sets using train_test_split.

```
from sklearn.model_selection import train_test_split

# Create a copy of the dataframe to avoid SettingWithCopyWarning
feature_df_copy = feature_df.copy()

# Drop rows with missing values in lagged sales features
feature_df_copy.dropna(subset=['Lagged_Sales_1M', 'Lagged_Sales_3M',␣
 ↪'Lagged_Sales_6M', 'Lagged_Sales_12M'], inplace=True)

# Separate features (X) and target (y)
X = feature_df_copy.drop('EV_Sales_Quantity', axis=1)
y = feature_df_copy['EV_Sales_Quantity']

# 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)
```

Model training

Train a linear regression model using the training data and make predictions on the test data.

```
from sklearn.linear_model import LinearRegression

# Instantiate a LinearRegression object
model = LinearRegression()

# Fit the model to the training data
model.fit(X_train, y_train)

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

# Print model coefficients and intercept (optional)
print("Model Coefficients:", model.coef_)
print("Model Intercept:", model.intercept_)
```

```
Model Coefficients: [ 5.87444164e-01  6.56060278e-02  4.09714483e-02
-6.24681423e-02
 -2.00177683e+01  3.46518572e+00 -7.06305989e+00  4.10658758e+00
  2.25142550e+01 -2.55025417e+01  5.75544746e-01  3.52529309e+01
 -1.27453295e+01 -6.07172728e+00  1.46446274e+01 -4.40048449e-01
 -4.48957638e+00 -1.17417848e+01 -1.08695469e+01  2.29891554e+01
  4.52246075e+00 -2.23947112e+01  6.04044927e+00  2.70747328e+01
 -1.95227104e+01 -1.18348754e+01 -1.29467466e+01 -8.89901209e+00


   -4.07266073e-01 -9.03798378e+00 -8.97817154e+00  1.16020049e+01
   -1.85504205e+01  2.25304875e+01 -2.54894583e+01  6.73116931e+01
   -6.47801306e+00  8.50637256e-01  2.51537072e+00  9.43346343e+00
    8.50248048e+00 -1.71355790e+01 -4.19571287e+01 -6.12747575e-01
    3.11036987e+01  1.47329366e+01  7.54951657e-14  1.17217728e+00
   -8.63902234e+00  8.88280968e-01 -1.11685868e+01  2.63569281e+00
    3.07256514e+00  5.28521384e+00  1.72730961e+00 -6.00225850e+01
    2.48047953e+02  5.10071312e+00 -3.26489153e+00  1.73680376e+01
    1.10763105e+01 -2.20530173e+00 -1.56982929e+01 -5.65173220e+00
    1.38864291e+01 -2.70705088e+00 -9.95270589e+00 -7.99186680e-01
   -7.22086262e-01  1.27650645e+02 -3.93426107e+01  1.41048628e+01
   -1.80384575e+01 -1.22413972e+01 -1.91651331e+01  6.21724894e-14
   -1.79233692e+01 -7.92368617e+00  4.26463689e+00  2.89213098e-14
    6.47209229e+00 -2.68095143e+01 -7.10542736e-15 -3.40797966e+01
   -4.22678423e+00 -8.01809850e+00 -1.54541322e+01  2.64538589e+01
    4.28860733e+00  5.67305385e+00 -4.00720676e+01 -3.23614749e+01
    5.85378522e+00 -1.14433399e+00 -4.10782519e-15  8.99280650e-15
   -4.79770611e+01 -4.28122929e+01 -6.19146205e+01 -1.11989811e+01
   -6.11071875e+00  6.94488370e+00  7.32536581e-01  4.08071022e-02
    6.66217526e+00  2.76588233e+01  0.00000000e+00  3.95908538e+00
    1.35392330e+00  8.68901775e+00  0.00000000e+00  9.49535435e+00
    3.53213939e+01 -4.38113555e+00 -1.43375970e+01 -2.60980157e+01
    1.42871590e+01 -5.47595712e-01  3.73590732e+00]
  Model Intercept: -28832.04575231855
```

Model evaluation

Evaluate the performance of the trained linear regression model using appropriate metrics.

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

# Calculate MAE, MSE, RMSE, and R-squared
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Print the calculated metrics
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R-squared (R2): {r2:.2f}")
```

```python
# Interpret the results
print("\nInterpretation:")
print("The MAE and RMSE represent the average prediction error of the model.")
print("A lower MAE and RMSE indicate better model performance.")
print("The R-squared value represents the proportion of variance in EV sales␣
 ↪explained by the model.")
print("A higher R-squared value indicates that the model is a better fit for␣
 ↪the data.")
```

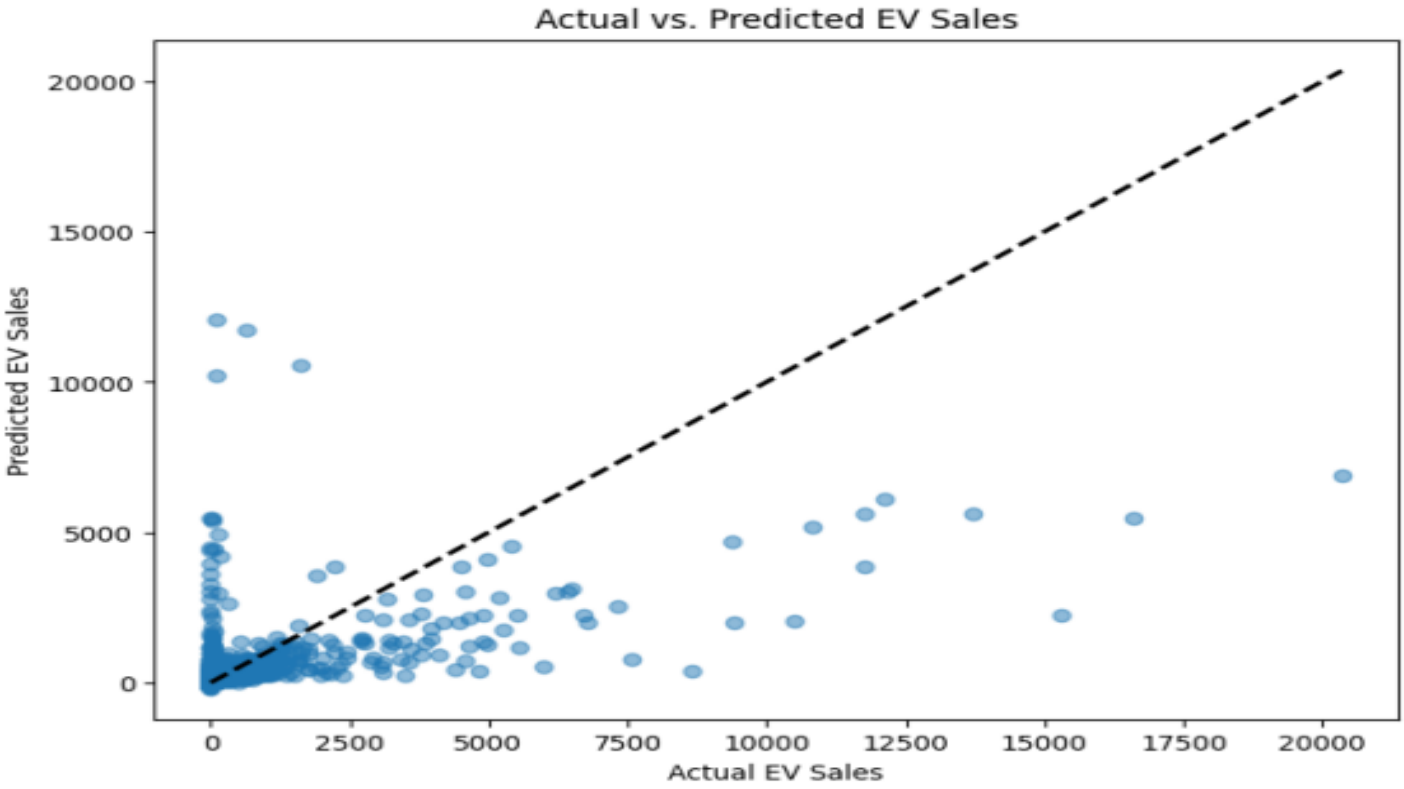
```
Mean Absolute Error (MAE): 74.17
Mean Squared Error (MSE): 143006.22
Root Mean Squared Error (RMSE): 378.16
R-squared (R2): 0.31

Interpretation:
The MAE and RMSE represent the average prediction error of the model.
A lower MAE and RMSE indicate better model performance.
The R-squared value represents the proportion of variance in EV sales explained
by the model.
A higher R-squared value indicates that the model is a better fit for the data.
```

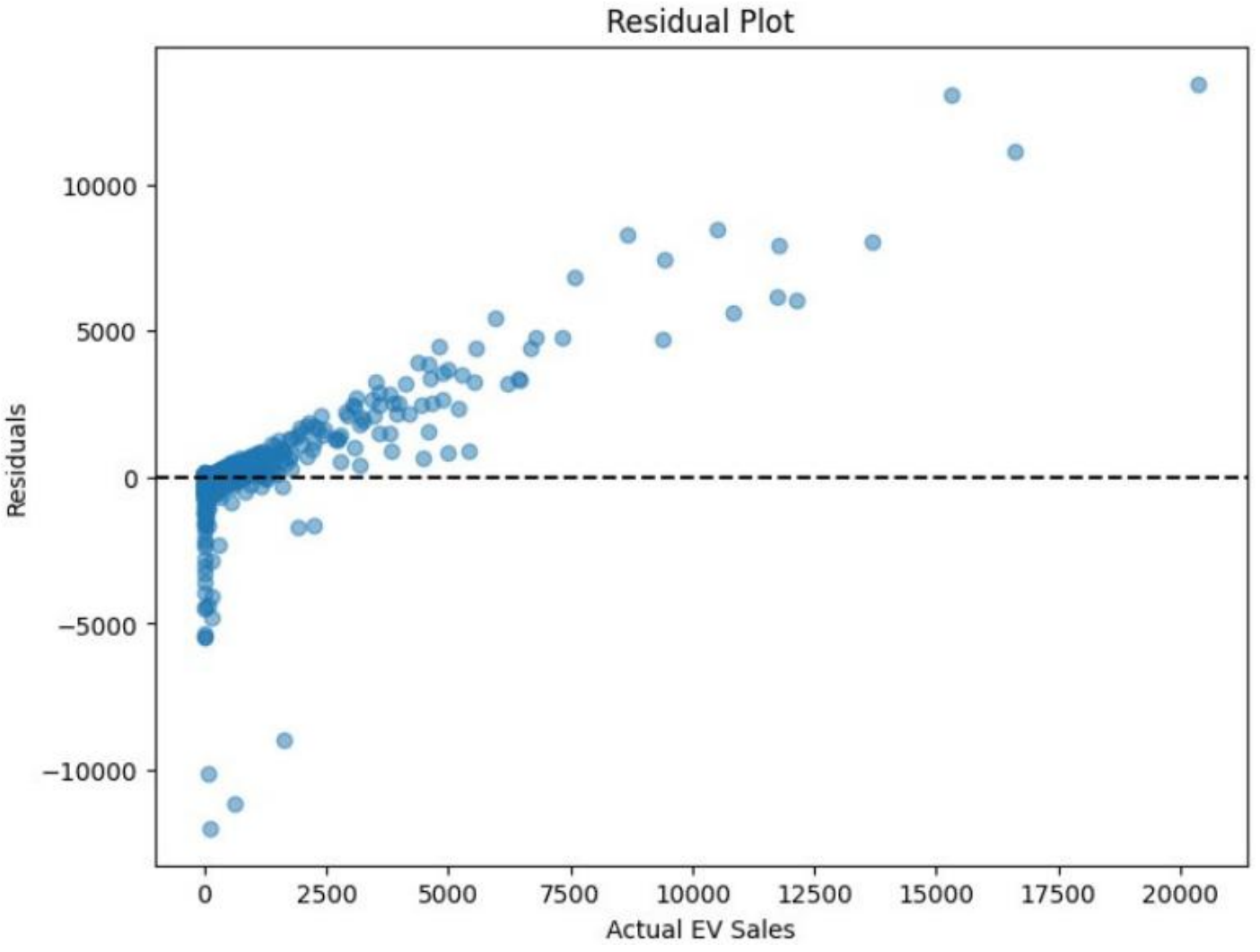
Data visualization

Visualize the predicted EV sales using the trained linear regression model by creating a scatter plot with the actual and predicted values, a diagonal line representing perfect predictions, and labels and a title.

```python
import matplotlib.pyplot as plt

# Create a scatter plot of actual vs. predicted EV sales
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel("Actual EV Sales")
plt.ylabel("Predicted EV Sales")
plt.title("Actual vs. Predicted EV Sales")

# Draw a diagonal line representing perfect predictions
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--',␣
 ↪lw=2)

plt.show()

# Calculate and plot residuals (optional)
residuals = y_test - y_pred
plt.figure(figsize=(8, 6))
```

```python
plt.scatter(y_test, residuals, alpha=0.5)
plt.xlabel("Actual EV Sales")
plt.ylabel("Residuals")
plt.title("Residual Plot")
plt.axhline(y=0, color='k', linestyle='--')
plt.show()
```


Actual vs. Predicted EV Sales
Predicted EV Sales
Actual EV Sales
0
5000
10000
15000
20000
0
2500
5000
7500
10000
12500
15000
17500
20000


Residual Plot
Residuals
Actual EV Sales
10000
5000
0
−5000
−10000
0
2500
5000
7500
10000
12500
15000
17500
20000

Q&A

- **What is the overall trend of EV sales in India?**

Based on the yearly analysis, EV sales in India show an overall upward trend, indicating growth in the EV market.

- **Which state has the highest EV sales?**

Uttar Pradesh has the highest EV sales in India.

- **Which state has the lowest EV sales?**

Sikkim has the lowest EV sales in India.

- **Which vehicle class has the highest sales?**

The 'M-Cycle/Scooter' vehicle class has the highest EV sales.

- **Which vehicle category has the highest sales?**

The '2-Wheelers' vehicle category has the highest EV sales.

- **How well does the linear regression model perform in predicting EV sales?**

The linear regression model has a moderate predictive power, explaining about 31% of the variance in EV sales (R-squared). The MAE and RMSE are relatively high, indicating room for improvement in accuracy.

Conclusion

The analysis and prediction of electric vehicle (EV) sales by state in India provide critical insights into the evolving landscape of sustainable transportation. As India moves towards achieving its green energy and carbon neutrality goals, the adoption of EVs is becoming a central focus of government policies, industry investments, and consumer awareness. Our study highlights significant trends in EV sales, emphasizing the disparities in adoption rates across different states. Leading states such as Maharashtra, Delhi, Karnataka, and Tamil Nadu showcase strong growth

due to proactive government policies, incentives, and robust charging infrastructure. In contrast, states with slower adoption rates indicate a need for better policy interventions, infrastructure development, and awareness programs. Through predictive modeling techniques, we forecast the future growth of EV sales based on historical data, economic factors, government policies, and consumer behavior patterns. The predictive analysis suggests a steady upward trajectory, with an expected acceleration in sales as technological advancements improve battery efficiency, reduce costs, and enhance charging accessibility. India's electric vehicle market presents a promising future, driven by technological advancements, favorable policies, and increasing environmental consciousness. By fostering innovation, expanding infrastructure, and ensuring equitable access to EVs across all states, India can achieve its ambitious sustainability goals while revolutionizing its transportation sector.