```
# Import necessary libraries
import pandas as pd
import numpy as np
```

Step 1: Data Collection

Start by loading the dataset. For this example, let's assume the dataset is in CSV format.

```
# Load the dataset
df = pd.read_csv('Electric Vehicle Sales by State in India.csv')
```

```
# Display the first few rows of the dataset
df.head()
```

	Year	Month_Name	Date	State	Vehicle_Class	Vehicle_Category	Vehicle_Type	EV_Sales_Quantity
0	2014.0	jan	1/1/2014	Andhra Pradesh	ADAPTED VEHICLE	Others	Others	0.0
1	2014.0	jan	1/1/2014	Andhra Pradesh	AGRICULTURAL TRACTOR	Others	Others	0.0
2	2014.0	jan	1/1/2014	Andhra Pradesh	AMBULANCE	Others	Others	0.0
3	2014.0	jan	1/1/2014	Andhra	ARTICULATED	Others	Others	0.0

Data Preprocessing

Handle missing values and convert the date column to a proper datetime format.

```
# Convert 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'])
```

```
df.head()
```

	Year	Month_Name	Date	State	Vehicle_Class	Vehicle_Category	Vehicle_Type	EV_Sales_Quantity
0	2014.0	jan	2014-01-01	Andhra Pradesh	ADAPTED VEHICLE	Others	Others	0.0
1	2014.0	jan	2014-01-01	Andhra Pradesh	AGRICULTURAL TRACTOR	Others	Others	0.0
2	2014.0	jan	2014-01-01	Andhra Pradesh	AMBULANCE	Others	Others	0.0
			2014-01-	Andhra	ARTICULATED			

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

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

```
# Fill missing values (if any) using median for numerical columns or mode for categorical columns
df['EV_Sales_Quantity'].fillna(df['EV_Sales_Quantity'].median(), inplace=True)
df.fillna(df.mode().iloc[0], inplace=True)
```

```
<ipython-input-10-c57d6aa50147>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series throug
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[
       df['EV_Sales_Quantity'].fillna(df['EV_Sales_Quantity'].median(), inplace=True)
```

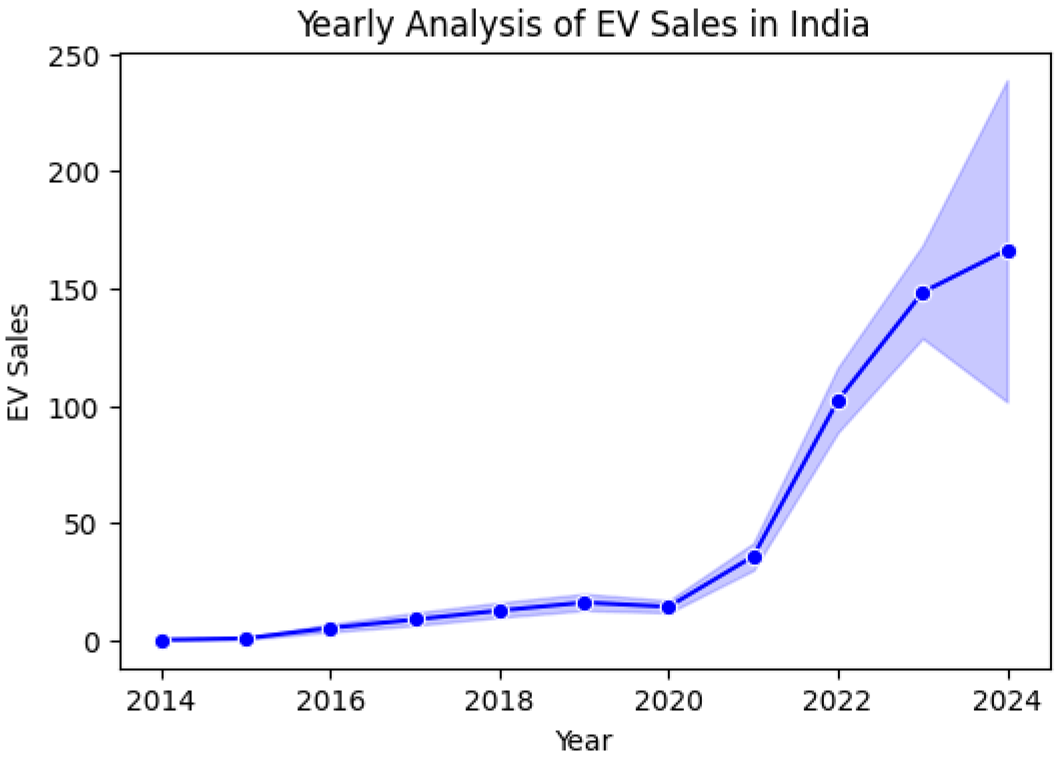
```
Start coding or generate with AI.
```

Exploratory Data Analysis (EDA)

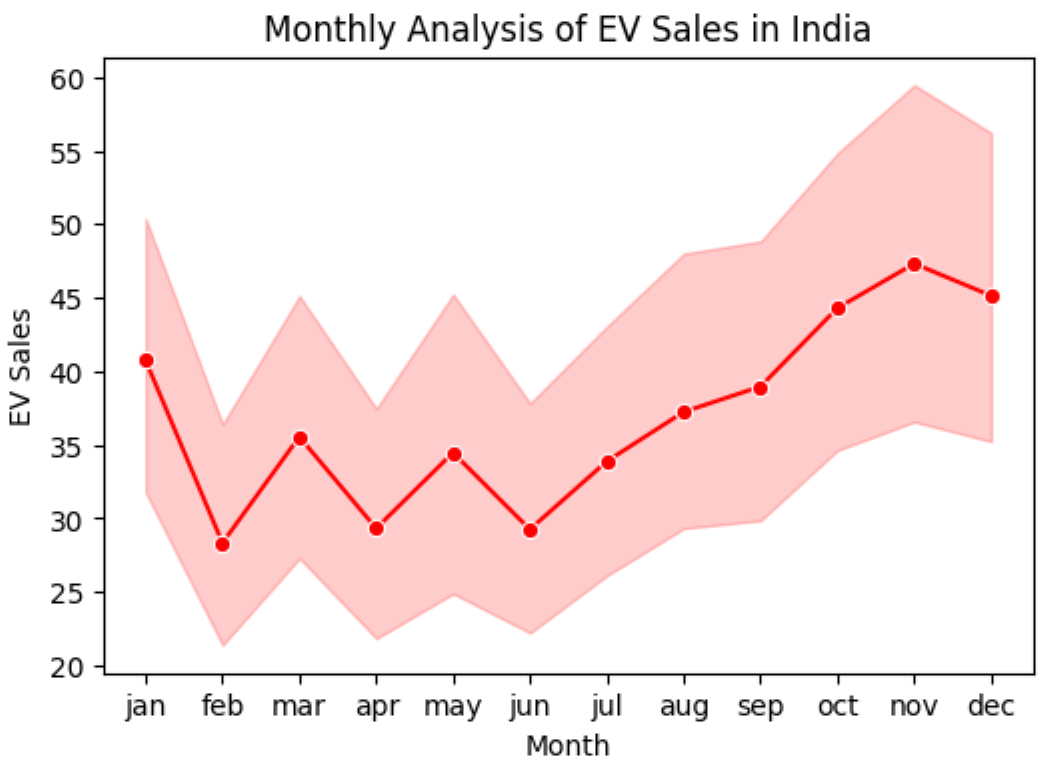
Visualize trends in EV sales over time, across states, and vehicle categories.

```
import matplotlib.pyplot as plt
import seaborn as sns
```

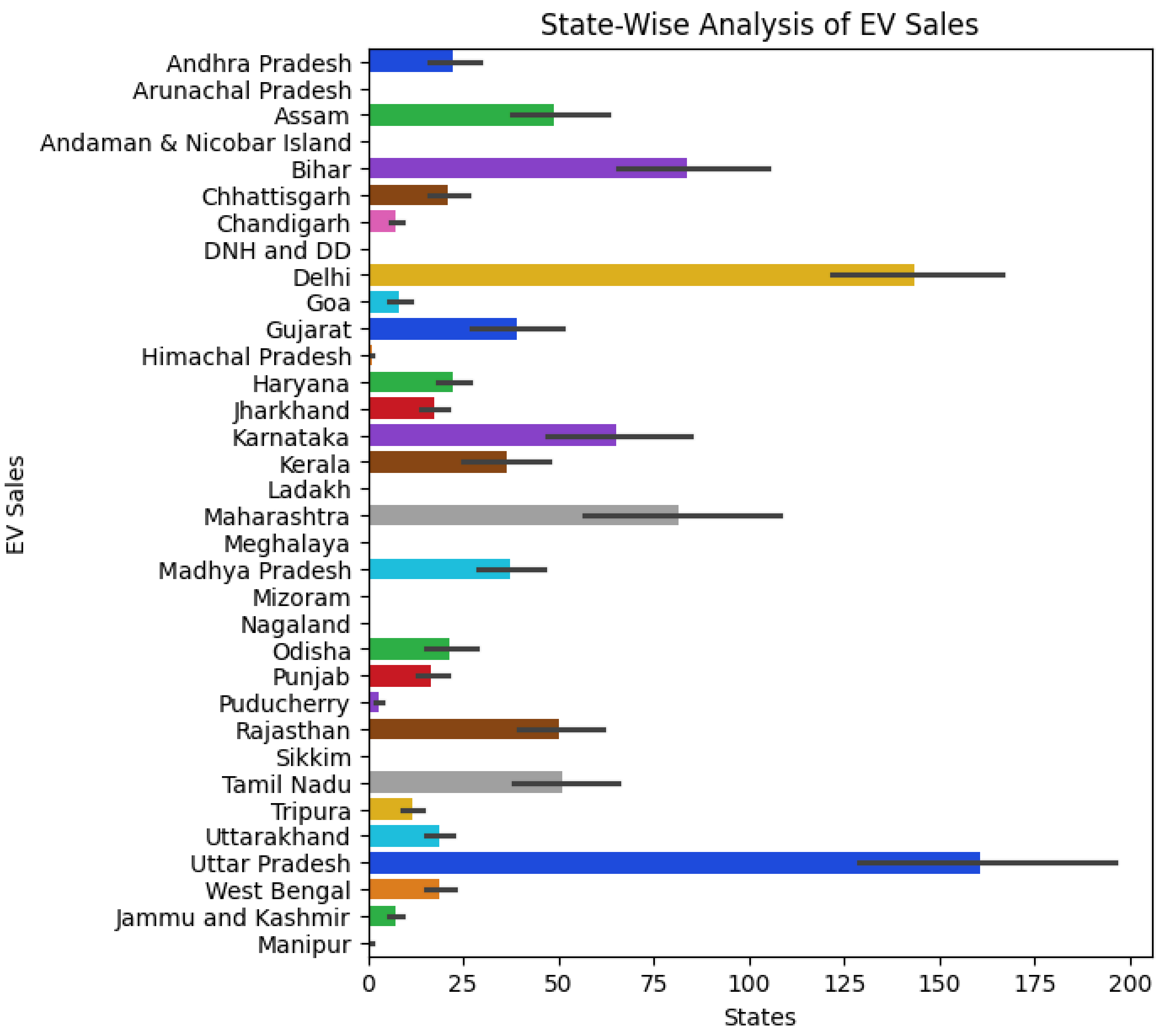
```
# Plot EV sales over the years
plt.figure(figsize=(6,4))
plt.title('Yearly Analysis of EV Sales in India')
sns.lineplot(x='Year', y='EV_Sales_Quantity', data=df,
marker='o', color='b')
plt.xlabel('Year')
plt.ylabel('EV Sales');
```



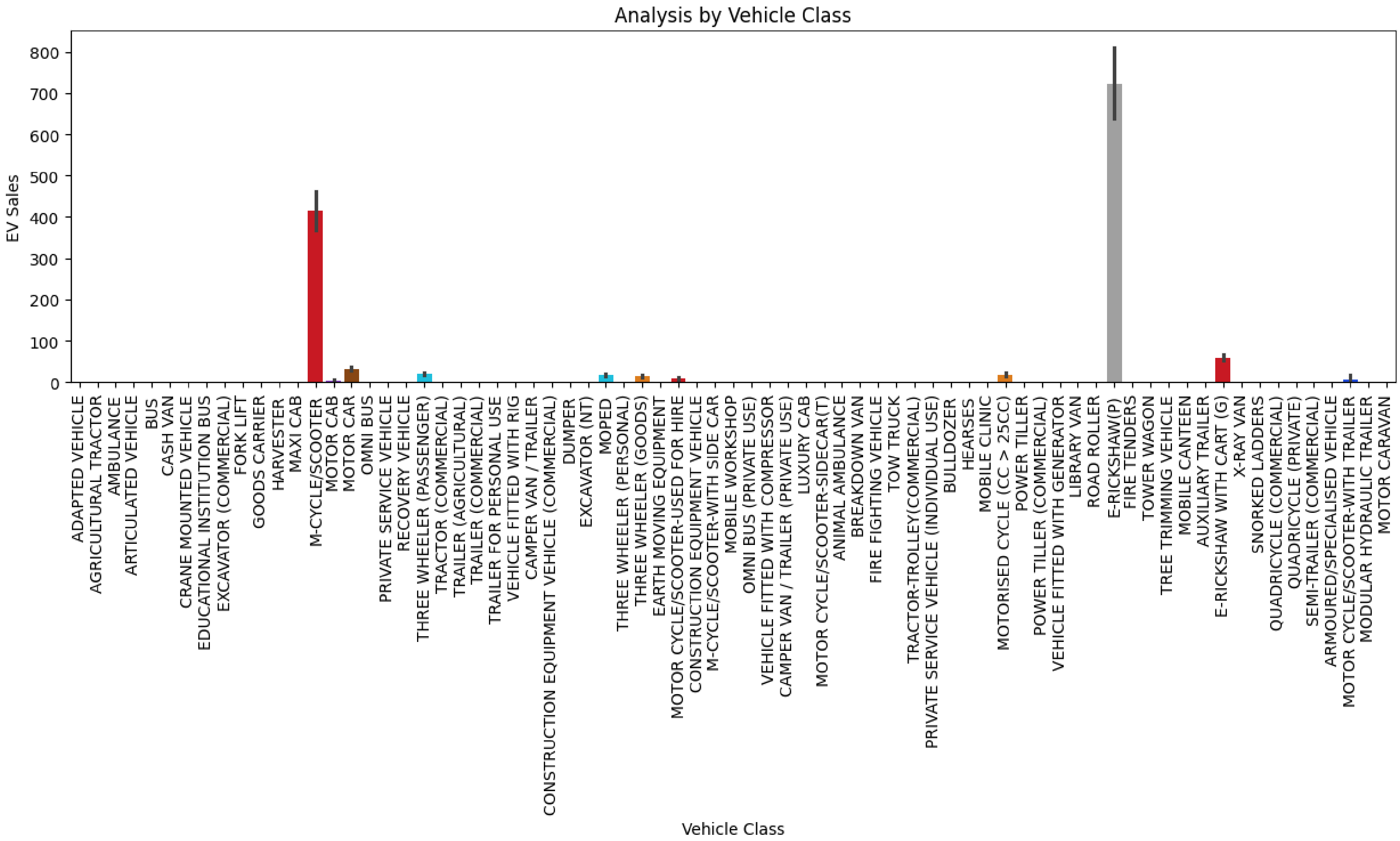
```
# Monthly Analysis of EV Sales in India
plt.figure(figsize=(6,4))
plt.title('Monthly Analysis of EV Sales in India')
sns.lineplot(x='Month_Name', y='EV_Sales_Quantity', data=df,
marker='o', color='r')
plt.xlabel('Month')
plt.ylabel('EV Sales');
```



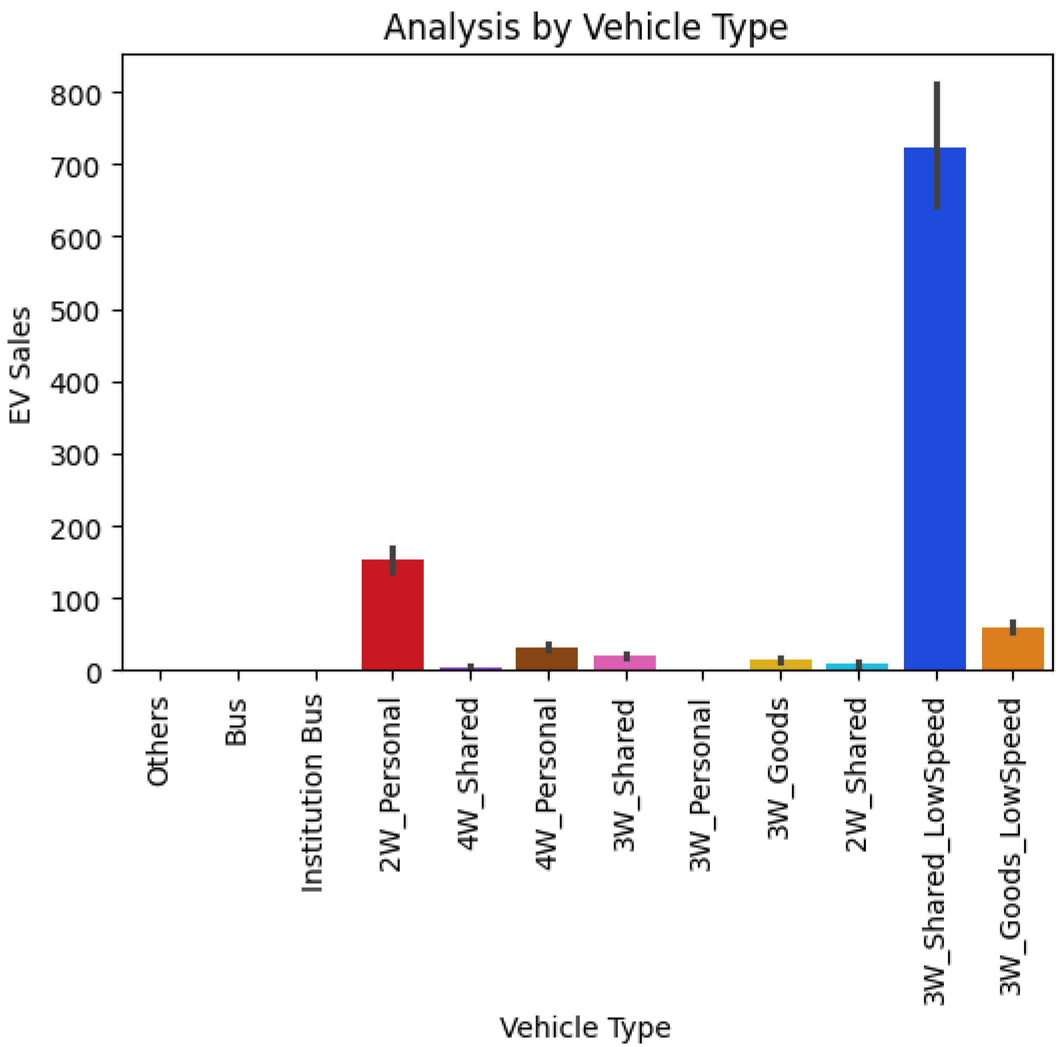
```
plt.figure(figsize=(6,7))
plt.title('State-Wise Analysis of EV Sales')
sns.barplot(y='State', x='EV_Sales_Quantity', data=df,
hue='State', palette='bright')
plt.xlabel('States')
plt.ylabel('EV Sales');
```



```
plt.figure(figsize=(15,4))
sns.barplot(x='Vehicle_Class', y='EV_Sales_Quantity',data=df,
hue='Vehicle_Class', palette='bright')
plt.title('Analysis by Vehicle Class')
plt.xlabel('Vehicle Class')
plt.ylabel('EV Sales')
plt.xticks(rotation=90);
```


Analysis by Vehicle Class
EV Sales
800
700
600
500
400
300
200
100
0
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
Vehicle Class

```
plt.figure(figsize=(6,4))
sns.barplot(x='Vehicle_Type', y='EV_Sales_Quantity',data=df,
hue='Vehicle_Type', palette='bright')
plt.title('Analysis by Vehicle Type')
plt.xlabel('Vehicle Type')
plt.ylabel('EV Sales')
plt.xticks(rotation=90);
```



Feature Engineering"

Create new features such as month and day from the Date column and encode categorical variables.

```
# Extract Month and Day from the Date column
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day


# Encode categorical variables using one-hot encoding
df_encoded = pd.get_dummies(df, columns=['State','Vehicle_Class', 'Vehicle_Category', 'Vehicle_Type'],drop_first=True)



# Drop unnecessary columns like Date, Month_Name (if already extracted into numerical values)
df_encoded.drop(['Date', 'Month_Name'], axis=1, inplace=True)


df.head()
```

	Year	Month_Name	Date	State	Vehicle_Class	Vehicle_Category	Vehicle_Type	EV_Sales_Quantity	Month	Day
0	2014.0	jan	2014-01-01	Andhra Pradesh	ADAPTED VEHICLE	Others	Others	0.0	1	1
1	2014.0	jan	2014-01-01	Andhra Pradesh	AGRICULTURAL TRACTOR	Others	Others	0.0	1	1
2	2014.0	jan	2014-01-01	Andhra Pradesh	AMBULANCE	Others	Others	0.0	1	1
			2014-	Andhra	ARTICULATED					

Start coding or generate with AI.

Modeling

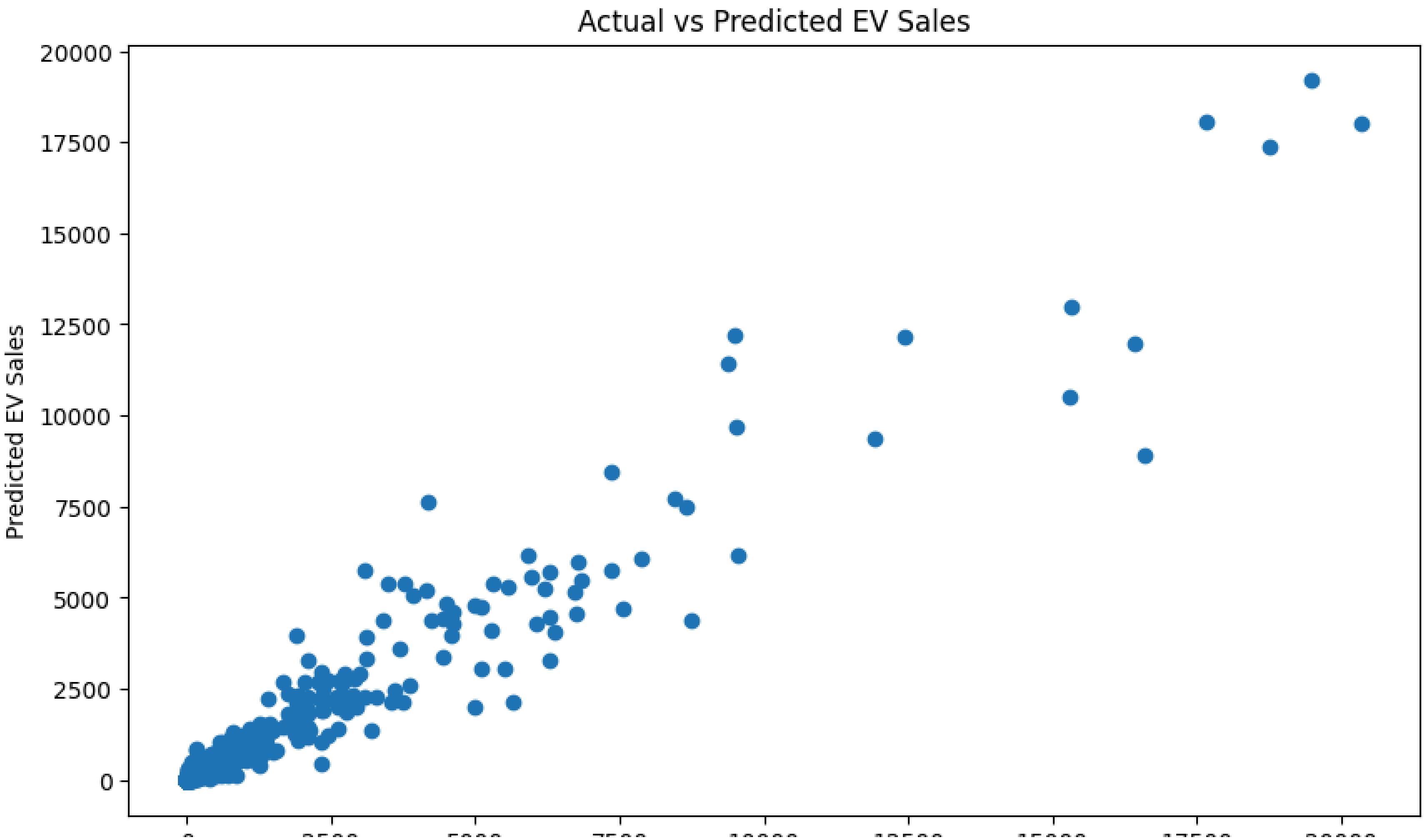
Use a regression model (e.g., Random Forest Regressor) to predict EV sales.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error


# Split the data into features and target variable
X = df_encoded.drop('EV_Sales_Quantity', axis=1)
y = df_encoded['EV_Sales_Quantity']


# Split the dataset 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)


# Instantiate the model
model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error: {rmse}')
```

```
Root Mean Squared Error: 130.58175227721011
```

Model Evaluation

```
# Plot actual vs predicted sales
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred)
plt.title('Actual vs Predicted EV Sales')
plt.xlabel('Actual EV Sales')
plt.ylabel('Predicted EV Sales')
plt.show()
```


Actual vs Predicted EV Sales
Predicted EV Sales