Title: ⚡ Electric Vehicle Analytics Dataset

Description (Detailed):

The Electric Vehicle Analytics Dataset provides comprehensive information about electric vehicles (EVs), capturing key attributes such as make, model, year of manufacture, vehicle type, range, battery capacity, charging speed, energy consumption, price, and adoption trends. It helps in analyzing the growth of EV adoption across regions, comparing performance between manufacturers, studying battery efficiency, and understanding consumer choices. This dataset can be applied to various domains including sustainability research, automobile industry forecasting, market trend analysis, and policy planning for green transportation.

👉 **It is especially useful for:**

Performance Analysis: Comparing vehicle range, battery size, and efficiency.

Market Trends: Tracking adoption rates, pricing trends, and regional popularity.

Sustainability Studies: Evaluating environmental impact and energy usage.

Import Library

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

Import Dataset

```
ev = pd.read_csv("electric_vehicle_analytics.csv")
ev.head()
```

	Vehicle_ID	Make	Model	Year	Region	Vehicle_Type	Battery_Capacity_kWh	Battery_Health_%	Range_km	Charging_Power_kW	...	Ma
0	1	Nissan	Leaf	2021	Asia	SUV	101.7	75.5	565	153.6	...	
1	2	Nissan	Leaf	2020	Australia	Sedan	30.1	99.8	157	157.2	...	
2	3	Hyundai	Kona Electric	2021	North America	SUV	118.5	84.0	677	173.6	...	
3	4	Audi	Q4 e-tron	2022	Europe	Hatchback	33.1	97.3	149	169.3	...	
4	5	Tesla	Model 3	2022	Australia	Truck	81.3	85.6	481	212.8	...	

5 rows × 25 columns

```
ev.tail()
```

	Vehicle_ID	Make	Model	Year	Region	Vehicle_Type	Battery_Capacity_kWh	Battery_Health_%	Range_km	Charging_Power_kW	..
2995	2996	Mercedes	EQS	2021	North America	SUV	57.2	84.0	239	102.2	.
2996	2997	Ford	Mustang Mach-E	2022	Europe	Hatchback	98.4	83.1	498	160.6	.
2997	2998	Kia	Niro EV	2024	Europe	Truck	35.1	82.1	189	18.1	.
2998	2999	Mercedes	EQC	2015	North America	Truck	69.4	98.4	336	94.7	.
2999	3000	Audi	Q4 e-tron	2023	North America	Hatchback	70.2	82.6	387	232.8	.

5 rows × 25 columns

```
ev.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 25 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Vehicle_ID                       3000 non-null   int64
 1   Make                             3000 non-null   object
 2   Model                            3000 non-null   object
 3   Year                             3000 non-null   int64
 4   Region                           3000 non-null   object
 5   Vehicle_Type                     3000 non-null   object
 6   Battery_Capacity_kWh             3000 non-null   float64
 7   Battery_Health_%                 3000 non-null   float64
 8   Range_km                         3000 non-null   int64
 9   Charging_Power_kW                3000 non-null   float64
 10  Charging_Time_hr                 3000 non-null   float64
 11  Charge_Cycles                    3000 non-null   int64
 12  Energy_Consumption_kWh_per_100km 3000 non-null   float64
 13  Mileage_km                       3000 non-null   int64
 14  Avg_Speed_kmh                    3000 non-null   float64
 15  Max_Speed_kmh                    3000 non-null   int64
 16  Acceleration_0_100_kmh_sec       3000 non-null   float64
 17  Temperature_C                    3000 non-null   float64
 18  Usage_Type                       3000 non-null   object
 19  CO2_Saved_tons                   3000 non-null   float64
```

```
 20  Maintenance_Cost_USD              3000 non-null   int64
 21  Insurance_Cost_USD                3000 non-null   int64
 22  Electricity_Cost_USD_per_kWh      3000 non-null   float64
 23  Monthly_Charging_Cost_USD         3000 non-null   float64
 24  Resale_Value_USD                  3000 non-null   int64
dtypes: float64(11), int64(9), object(5)
memory usage: 586.1+ KB
```

```
ev.describe()
```

	Vehicle_ID	Year	Battery_Capacity_kWh	Battery_Health_%	Range_km	Charging_Power_kW	Charging_Time_hr	Charge_Cycles
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
mean	1500.500000	2019.499667	74.810100	85.030000	374.414667	129.301000	1.203570	1107.009667
std	866.169729	2.848047	25.734079	8.589526	137.184112	68.742745	1.421866	510.834590
min	1.000000	2015.000000	30.000000	70.000000	121.000000	11.100000	0.140000	200.000000
25%	750.750000	2017.000000	53.000000	77.775000	260.000000	70.900000	0.460000	674.750000
50%	1500.500000	2020.000000	74.850000	85.250000	371.000000	126.700000	0.720000	1116.000000
75%	2250.250000	2022.000000	96.900000	92.300000	476.250000	187.975000	1.292500	1535.250000
max	3000.000000	2024.000000	120.000000	100.000000	713.000000	250.000000	12.140000	1997.000000

```
ev.isnull().sum()
```

	0
Vehicle_ID	0
Make	0
Model	0
Year	0
Region	0
Vehicle_Type	0
Battery_Capacity_kWh	0
Battery_Health_%	0
Range_km	0
Charging_Power_kW	0
Charging_Time_hr	0
Charge_Cycles	0
Energy_Consumption_kWh_per_100km	0
Mileage_km	0
Avg_Speed_kmh	0
Max_Speed_kmh	0
Acceleration_0_100_kmh_sec	0
Temperature_C	0
Usage_Type	0
CO2_Saved_tons	0
Maintenance_Cost_USD	0
Insurance_Cost_USD	0
Electricity_Cost_USD_per_kWh	0
Monthly_Charging_Cost_USD	0
Resale_Value_USD	0

dtype: int64

```
ev.nunique()
```

	0
Vehicle_ID	3000
Make	10
Model	23
Year	10
Region	4
Vehicle_Type	4
Battery_Capacity_kWh	878
Battery_Health_%	301
Range_km	547
Charging_Power_kW	1708
Charging_Time_hr	452
Charge_Cycles	1455
Energy_Consumption_kWh_per_100km	1165
Mileage_km	2979
Avg_Speed_kmh	693
Max_Speed_kmh	120
Acceleration_0_100_kmh_sec	644
Temperature_C	501
Usage_Type	3
CO2_Saved_tons	1895
Maintenance_Cost_USD	1481
Insurance_Cost_USD	1547
Electricity_Cost_USD_per_kWh	28
Monthly_Charging_Cost_USD	2953
Resale_Value_USD	2791

dtype: int64

```
ev.duplicated().sum()
```

```
np.int64(0)
```

```
ev.dtypes
```

	0
Vehicle_ID	int64
Make	object
Model	object
Year	int64
Region	object
Vehicle_Type	object
Battery_Capacity_kWh	float64
Battery_Health_%	float64
Range_km	int64
Charging_Power_kW	float64
Charging_Time_hr	float64
Charge_Cycles	int64
Energy_Consumption_kWh_per_100km	float64
Mileage_km	int64
Avg_Speed_kmh	float64
Max_Speed_kmh	int64
Acceleration_0_100_kmh_sec	float64
Temperature_C	float64
Usage_Type	object
CO2_Saved_tons	float64
Maintenance_Cost_USD	int64
Insurance_Cost_USD	int64
Electricity_Cost_USD_per_kWh	float64
Monthly_Charging_Cost_USD	float64
Resale_Value_USD	int64

dtype: object

```
ev.shape
```

```
(3000, 25)
```

```
ev.columns
```

```
Index(['Vehicle_ID', 'Make', 'Model', 'Year', 'Region', 'Vehicle_Type',
       'Battery_Capacity_kWh', 'Battery_Health_%', 'Range_km',
       'Charging_Power_kW', 'Charging_Time_hr', 'Charge_Cycles',
       'Energy_Consumption_kWh_per_100km', 'Mileage_km', 'Avg_Speed_kmh',
       'Max_Speed_kmh', 'Acceleration_0_100_kmh_sec', 'Temperature_C',
       'Usage_Type', 'CO2_Saved_tons', 'Maintenance_Cost_USD',
       'Insurance_Cost_USD', 'Electricity_Cost_USD_per_kWh',
       'Monthly_Charging_Cost_USD', 'Resale_Value_USD'],
      dtype='object')
```

Data Analysis

```
# 1) Energy Efficiency: Range per kWh by model
ev["Range_per_kWh"] = ev["Range_km"] / ev["Battery_Capacity_kWh"]
efficiency = (
    ev.groupby(["Make", "Model"], as_index=False)
      .agg(Avg_Range_per_kWh=("Range_per_kWh", "mean"))
      .sort_values("Avg_Range_per_kWh", ascending=False)
)
print("\n Top 10 Models by Range per kWh (Efficiency)")
print(efficiency.head(10))
```

```
 Top 10 Models by Range per kWh (Efficiency)
          Make            Model  Avg_Range_per_kWh
10     Hyundai    Kona Electric           5.048726
1         Audi          e-tron           5.044854
8         Ford   Mustang Mach-E           5.038327
5    Chevrolet         Bolt EUV           5.035450
16      Nissan             Leaf           5.030865
7         Ford  F-150 Lightning           5.030233
0         Audi        Q4 e-tron           5.013994
9      Hyundai          Ioniq 5           5.010676
22  Volkswagen             ID.4           5.008893
2          BMW               i3           5.004751
```

```
# 2) Charging Cost across Regions & Vehicle Types
region_cost = df.groupby("Region")["Monthly_Charging_Cost_USD"].mean()
vehicle_cost = df.groupby("Vehicle_Type")["Monthly_Charging_Cost_USD"].mean()
print("\n Charging Cost by Region")
print(region_cost)
print("\n Charging Cost by Vehicle Type")
print(vehicle_cost)
```

```
 Charging Cost by Region
Region
Asia             429.954083
Australia        405.581045
Europe           429.933992
North America    410.965486
Name: Monthly_Charging_Cost_USD, dtype: float64

 Charging Cost by Vehicle Type
Vehicle_Type
Hatchback    425.649638
SUV          406.122889
Sedan        429.416707
Truck        414.120635
Name: Monthly_Charging_Cost_USD, dtype: float64
```

```
# 3) Battery Health vs Mileage & Charge Cycles
corr_mileage = ev["Battery_Health_%"].corr(ev["Mileage_km"])
corr_cycles = ev["Battery_Health_%"].corr(ev["Charge_Cycles"])
print("\n Correlation Battery Health vs Mileage:", corr_mileage)
print(" Correlation Battery Health vs Charge Cycles:", corr_cycles)
```

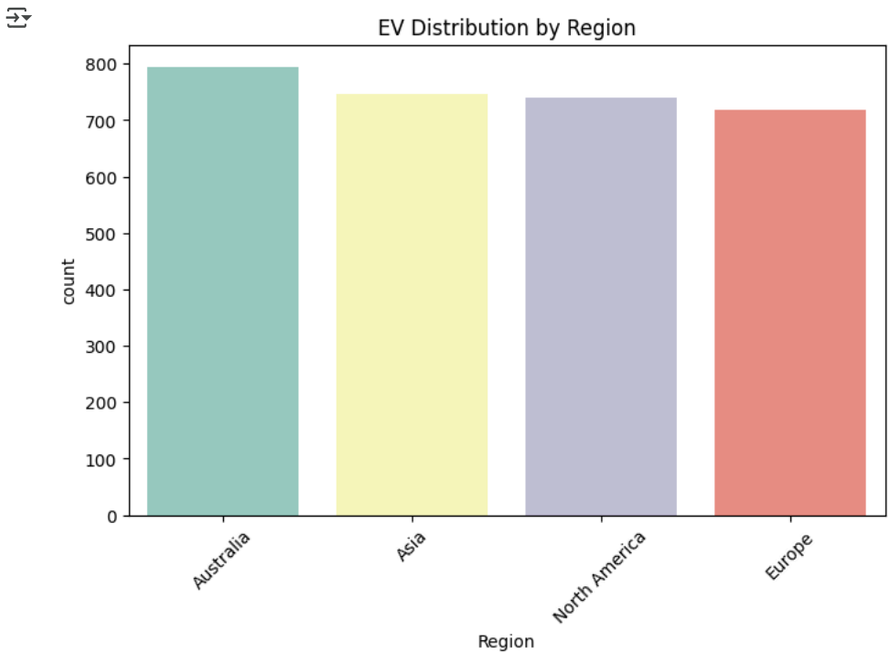
```
 Correlation Battery Health vs Mileage: 0.02885496858459314
 Correlation Battery Health vs Charge Cycles: -0.017264067741648154
```

```
# 4) CO2 Savings by Region and Usage Type
co2_region = ev.groupby("Region")["CO2_Saved_tons"].sum()
co2_usage = ev.groupby("Usage_Type")["CO2_Saved_tons"].sum()

print("\n Total CO2 Saved by Region")
print(co2_region)
print("\n Total CO2 Saved by Usage Type")
print(co2_usage)
```

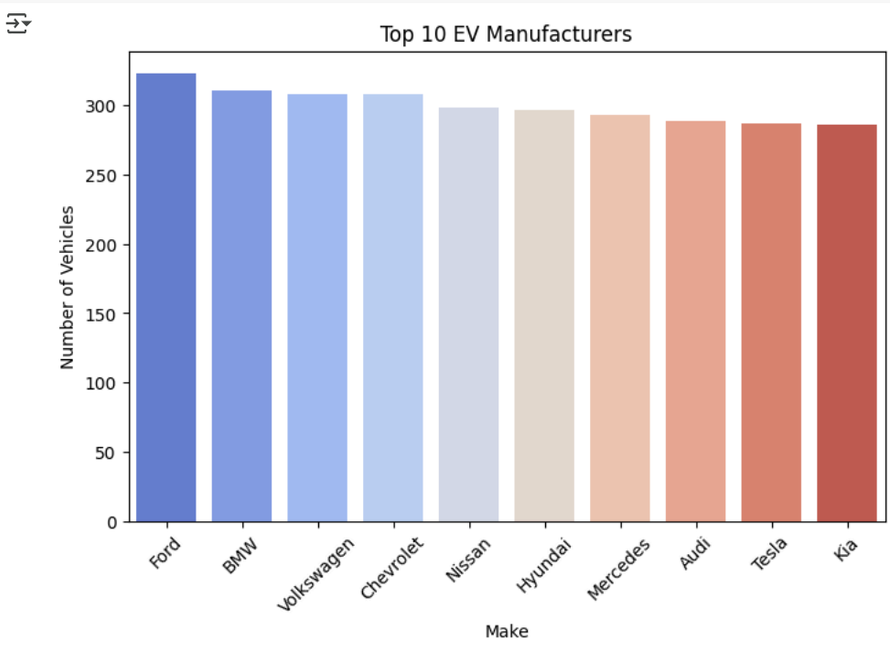
```
 Total CO2 Saved by Region
Region
Asia             11277.96
Australia        11817.96
Europe           11005.78
North America    10973.79
Name: CO2_Saved_tons, dtype: float64

 Total CO2 Saved by Usage Type
Usage_Type
Commercial    15016.77
Fleet         14323.77
Personal      15734.95
Name: CO2_Saved_tons, dtype: float64
```

Data Visualizations

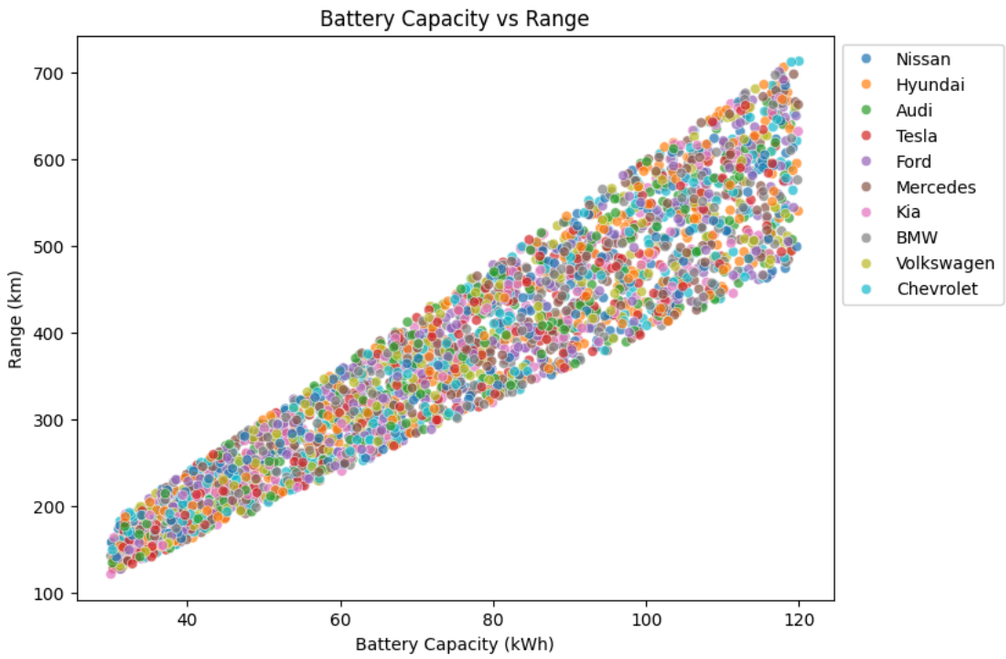
```
# 1. Distribution of EVs by Region
plt.figure(figsize=(8,5))
sns.countplot(data=ev, x="Region", order=ev["Region"].value_counts().index, palette="Set3")
plt.title("EV Distribution by Region")
plt.xticks(rotation=45)
plt.show()
```



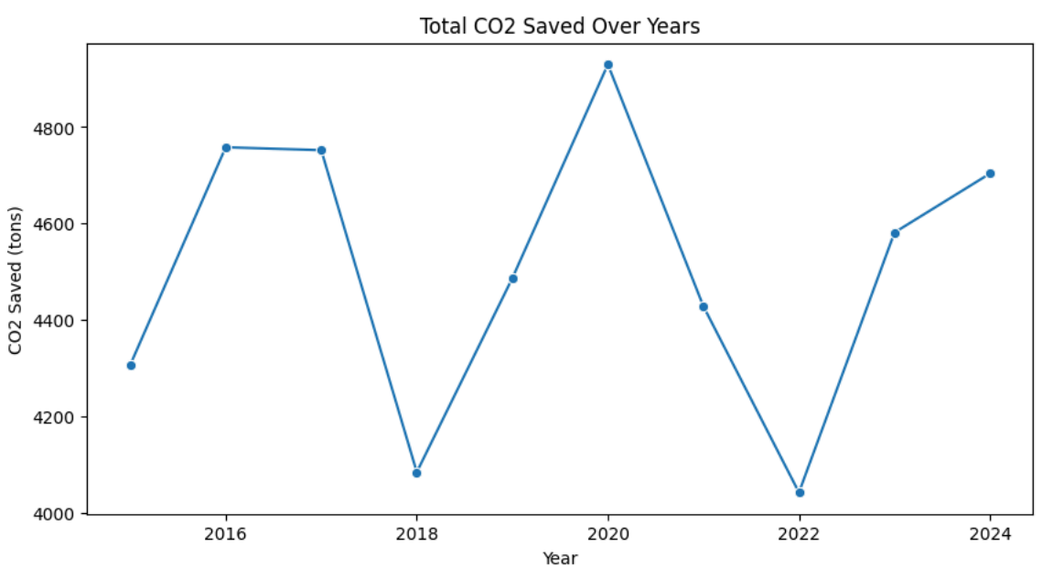

```
# 2. Top EV Manufacturers
plt.figure(figsize=(8,5))
top_makes = ev["Make"].value_counts().head(10)
sns.barplot(x=top_makes.index, palette="coolwarm", y=top_makes.values)
plt.title("Top 10 EV Manufacturers")
plt.xticks(rotation=45)
plt.ylabel("Number of Vehicles")
plt.show()
```



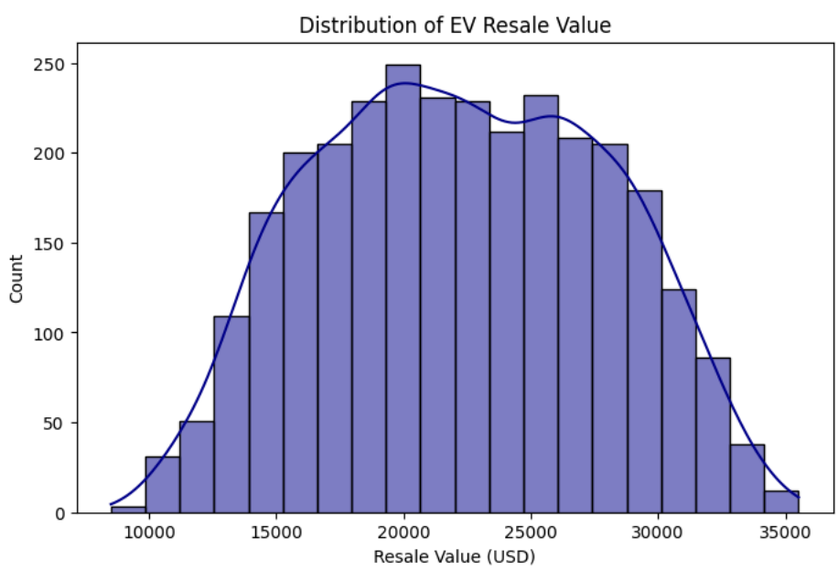

```
# 3. Vehicle Range vs Battery Capacity
plt.figure(figsize=(8,6))
sns.scatterplot(data=ev, x="Battery_Capacity_kWh", y="Range_km", hue="Make", alpha=0.7)
plt.title("Battery Capacity vs Range")
plt.xlabel("Battery Capacity (kWh)")
plt.ylabel("Range (km)")
plt.legend(bbox_to_anchor=(1,1))
plt.show()
```




```
# 7. CO2 Savings by Year
plt.figure(figsize=(10,5))
co2_year = ev.groupby("Year")["CO2_Saved_tons"].sum().reset_index()
sns.lineplot(data=co2_year, x="Year", y="CO2_Saved_tons", marker="o")
plt.title("Total CO2 Saved Over Years")
plt.ylabel("CO2 Saved (tons)")
plt.show()
```




```
# 9. Resale Value Distribution
plt.figure(figsize=(8,5))
sns.histplot(ev["Resale_Value_USD"], bins=20, kde=True, color="darkblue")
plt.title("Distribution of EV Resale Value")
plt.xlabel("Resale Value (USD)")
plt.show()
```



```
# 10. Monthly Charging Cost by Region
plt.figure(figsize=(8,5))
sns.boxplot(data=ev, palette="viridis",x="Region", y="Monthly_Charging_Cost_USD")
plt.title("Monthly Charging Cost Distribution by Region")
plt.xticks(rotation=45)
plt.ylabel("Monthly Charging Cost (USD)")
plt.show()
```