ev-eda

October 10, 2024

[81]:

# Epanagalla Ramakrishna

## ID : IN9240255

**import pandas as pd import numpy as np import seaborn as sns**

**import matplotlib.pyplot as plt**

[83]:

df = pd.read\_csv("dataset.csv")

[85]:

df.head()

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [85]: | VIN (1-10) | County | City | State | Postal Code | Model Year | Make | \ |
|  | 0 JTMEB3FV6N | Monroe | Key West | FL | 33040 | 2022 | TOYOTA |  |
|  | 1 1G1RD6E45D | Clark | Laughlin | NV | 89029 | 2013 | CHEVROLET |  |
|  | 2 JN1AZ0CP8B | Yakima | Yakima | WA | 98901 | 2011 | NISSAN |  |
|  | 3 1G1FW6S08H | Skagit | Concrete | WA | 98237 | 2017 | CHEVROLET |  |
|  | 4 3FA6P0SU1K | Snohomish | Everett | WA | 98201 | 2019 | FORD |  |
| Model   1. RAV4 PRIME 2. VOLT 3. LEAF 4. BOLT EV 5. FUSION | | Electric Vehicle Type Plug-in Hybrid Electric Vehicle (PHEV) Plug-in Hybrid Electric Vehicle (PHEV) Battery Electric Vehicle (BEV)  Battery Electric Vehicle (BEV)  Plug-in Hybrid Electric Vehicle (PHEV) | | | | \ | | |

Clean Alternative Fuel Vehicle (CAFV) Eligibility Electric Range \

1. Clean Alternative Fuel Vehicle Eligible 42
2. Clean Alternative Fuel Vehicle Eligible 38
3. Clean Alternative Fuel Vehicle Eligible 73
4. Clean Alternative Fuel Vehicle Eligible 238
5. Not eligible due to low battery range 26

Base MSRP Legislative District DOL Vehicle ID \

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | NaN | 198968248 |
| 1 | 0 | NaN | 5204412 |
| 2 | 0 | 15.0 | 218972519 |

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 0 | 39.0 | 186750406 |
| 4 | 0 | 38.0 | 2006714 |

[87]:

df.describe()

Vehicle Location Electric Utility 2020 Census Tract

0 POINT (-81.80023 24.5545) NaN 12087972100

1 POINT (-114.57245 35.16815) NaN 32003005702

2 POINT (-120.50721 46.60448) PACIFICORP 53077001602

3 POINT (-121.7515 48.53892) PUGET SOUND ENERGY INC 53057951101

4 POINT (-122.20596 47.97659) PUGET SOUND ENERGY INC 53061041500

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [87]: | Postal Code | Model Year | Electric Range | Base MSRP | \ |
| count | 112634.000000 | 112634.000000 | 112634.000000 | 112634.000000 |  |
| mean | 98156.226850 | 2019.003365 | 87.812987 | 1793.439681 |  |
| std | 2648.733064 | 2.892364 | 102.334216 | 10783.753486 |  |
| min | 1730.000000 | 1997.000000 | 0.000000 | 0.000000 |  |
| 25% | 98052.000000 | 2017.000000 | 0.000000 | 0.000000 |  |
| 50% | 98119.000000 | 2020.000000 | 32.000000 | 0.000000 |  |
| 75% | 98370.000000 | 2022.000000 | 208.000000 | 0.000000 |  |
| max | 99701.000000 | 2023.000000 | 337.000000 | 845000.000000 |  |

Legislative District DOL Vehicle ID 2020 Census Tract

|  |  |  |  |
| --- | --- | --- | --- |
| count | 112348.000000 | 1.126340e+05 | 1.126340e+05 |
| mean | 29.805604 | 1.994567e+08 | 5.296650e+10 |
| std | 14.700545 | 9.398427e+07 | 1.699104e+09 |
| min | 1.000000 | 4.777000e+03 | 1.101001e+09 |
| 25% | 18.000000 | 1.484142e+08 | 5.303301e+10 |
| 50% | 34.000000 | 1.923896e+08 | 5.303303e+10 |
| 75% | 43.000000 | 2.191899e+08 | 5.305307e+10 |
| max | 49.000000 | 4.792548e+08 | 5.603300e+10 |

[89]:

df.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 112634 entries, 0 to 112633 Data columns (total 17 columns):

# Column Non-Null Count Dtype

1. VIN (1-10) 112634 non-null object
2. County 112634 non-null object
3. City 112634 non-null object
4. State 112634 non-null object
5. Postal Code 112634 non-null int64
6. Model Year 112634 non-null int64
7. Make 112634 non-null object
8. Model 112614 non-null object

[91]:

1. Electric Vehicle Type 112634 non-null object
2. Clean Alternative Fuel Vehicle (CAFV) Eligibility 112634 non-null object
3. Electric Range 112634 non-null int64
4. Base MSRP 112634 non-null int64
5. Legislative District 112348 non-null float64
6. DOL Vehicle ID 112634 non-null int64
7. Vehicle Location 112610 non-null object
8. Electric Utility 112191 non-null object
9. 2020 Census Tract 112634 non-null int64 dtypes: float64(1), int64(6), object(10)

memory usage: 14.6+ MB

df.shape

[91]: (112634, 17)

[93]:

df.columns

[93]: Index(['VIN (1-10)', 'County', 'City', 'State', 'Postal Code', 'Model Year', 'Make', 'Model', 'Electric Vehicle Type',

'Clean Alternative Fuel Vehicle (CAFV) Eligibility', 'Electric Range', 'Base MSRP', 'Legislative District', 'DOL Vehicle ID',

'Vehicle Location', 'Electric Utility', '2020 Census Tract'], dtype='object')

[95]:

df.columns = df.columns.str.replace(' ', '\_') df.columns

[95]: Index(['VIN\_(1-10)', 'County', 'City', 'State', 'Postal\_Code', 'Model\_Year', 'Make', 'Model', 'Electric\_Vehicle\_Type', 'Clean\_Alternative\_Fuel\_Vehicle\_(CAFV)\_Eligibility', 'Electric\_Range', 'Base\_MSRP', 'Legislative\_District', 'DOL\_Vehicle\_ID', 'Vehicle\_Location', 'Electric\_Utility', '2020\_Census\_Tract'], dtype='object')

[97]:

df.rename(columns={'Clean\_Alternative\_Fuel\_Vehicle\_(CAFV)\_Eligibility':

𝗌'CAFV\_Eligibility'}, inplace=**True**) df.columns

[97]: Index(['VIN\_(1-10)', 'County', 'City', 'State', 'Postal\_Code', 'Model\_Year', 'Make', 'Model', 'Electric\_Vehicle\_Type', 'CAFV\_Eligibility', 'Electric\_Range', 'Base\_MSRP', 'Legislative\_District', 'DOL\_Vehicle\_ID', 'Vehicle\_Location', 'Electric\_Utility', '2020\_Census\_Tract'], dtype='object')

[99]:

print(df.isnull().sum())

VIN\_(1-10) 0

[101]:

County 0

City 0

State 0

Postal\_Code 0

Model\_Year 0

Make 0

Model 20

Electric\_Vehicle\_Type 0

CAFV\_Eligibility 0

Electric\_Range 0

Base\_MSRP 0

Legislative\_District 286

DOL\_Vehicle\_ID 0

Vehicle\_Location 24

Electric\_Utility 443

2020\_Census\_Tract 0

dtype: int64

df\_dropna = df.dropna()

df\_dropna.info()

<class 'pandas.core.frame.DataFrame'> Index: 112152 entries, 2 to 112633 Data columns (total 17 columns):

# Column Non-Null Count Dtype

|  |  |  |  |
| --- | --- | --- | --- |
| 0 VIN\_(1-10) | 112152 | non-null | object |
| 1 County | 112152 | non-null | object |
| 2 City | 112152 | non-null | object |
| 3 State | 112152 | non-null | object |
| 4 Postal\_Code | 112152 | non-null | int64 |
| 5 Model\_Year | 112152 | non-null | int64 |
| 6 Make | 112152 | non-null | object |
| 7 Model | 112152 | non-null | object |
| 8 Electric\_Vehicle\_Type | 112152 | non-null | object |
| 9 CAFV\_Eligibility | 112152 | non-null | object |
| 10 Electric\_Range | 112152 | non-null | int64 |
| 11 Base\_MSRP | 112152 | non-null | int64 |
| 12 Legislative\_District | 112152 | non-null | float64 |
| 13 DOL\_Vehicle\_ID | 112152 | non-null | int64 |
| 14 Vehicle\_Location | 112152 | non-null | object |
| 15 Electric\_Utility | 112152 | non-null | object |
| 16 2020\_Census\_Tract | 112152 | non-null | int64 |

dtypes: float64(1), int64(6), object(10) memory usage: 15.4+ MB

[ ]:

[106]:

# Task - 1

## Non-Visual Univariate Analysis

numerical\_columns = ['Postal\_Code', 'Model\_Year', 'Electric\_Range',␣

𝗌'Base\_MSRP', 'Legislative\_District', 'DOL\_Vehicle\_ID', '2020\_Census\_Tract']

categorical\_columns = ['VIN\_(1-10)', 'County', 'City', 'State', 'Make',␣

𝗌'Model', 'Electric\_Vehicle\_Type', 'CAFV\_Eligibility', 'Vehicle\_Location',␣

𝗌'Electric\_Utility']

discrete\_df = df.select\_dtypes(include=['object']) numerical\_df = df.select\_dtypes(include=['int64', 'float64'])

[108]:

**def** discrete\_univariate\_analysis(discrete\_data):

**for** col\_name **in** discrete\_data:

print("-"\*10, col\_name, "-"\*10) print(discrete\_data[col\_name].agg(['count', 'nunique', 'unique'])) print('Value Counts: **\n**', discrete\_data[col\_name].value\_counts()) print()

[110]:

discrete\_univariate\_analysis(discrete\_df)

VIN\_(1-10)

count 112634

nunique 7548

unique [JTMEB3FV6N, 1G1RD6E45D, JN1AZ0CP8B, 1G1FW6S08…

Name: VIN\_(1-10), dtype: object Value Counts:

VIN\_(1-10)

|  |  |
| --- | --- |
| 5YJYGDEE9M | 472 |
| 5YJYGDEE0M | 465 |
| 5YJYGDEE8M | 448 |
| 5YJYGDEE7M | 448 |
| 5YJYGDEE2M | 437  … |
| WA1LAAGE9M | 1 |
| 5UXKT0C50H | 1 |
| 5YJYGAED3M | 1 |
| WDC0G5DBXL | 1 |
| YV4ED3GM0P | 1 |
| Name: count, | Length: 7548, dtype: int64 |

County

count 112634

nunique 165

unique [Monroe, Clark, Yakima, Skagit, Snohomish, Isl… Name: County, dtype: object

Value Counts:

County

|  |  |
| --- | --- |
| King | 59000 |
| Snohomish | 12434 |
| Pierce | 8535 |
| Clark | 6689 |
| Thurston | 4126  … |
| Pinal | 1 |
| Elmore | 1 |
| Portsmouth | 1 |
| Kings | 1 |
| Kootenai | 1 |
| Name: count, | Length: 165, dtype: int64 |

City

count 112634

nunique 629

unique [Key West, Laughlin, Yakima, Concrete, Everett… Name: City, dtype: object

Value Counts:

City

|  |  |
| --- | --- |
| Seattle | 20305 |
| Bellevue | 5921 |
| Redmond | 4201 |
| Vancouver | 4013 |
| Kirkland | 3598  … |
| Hartline | 1 |
| Gaithersburg | 1 |
| El Paso | 1 |
| Klickitat | 1 |
| Worley | 1 |
| Name: count, | Length: 629, dtype: int64 |

State

count 112634

nunique 45

unique [FL, NV, WA, IL, NY, VA, OK, KS, CA, NE, MD, C…

Name: State, dtype: object Value Counts:

State

WA 112348

|  |  |  |
| --- | --- | --- |
| CA | 76 |  |
| VA | 36 |  |
| MD | 26 |  |
| TX | 14 |  |
| CO | 9 |  |
| NV | 8 |  |
| GA | 7 |  |
| NC | 7 |  |
| CT | 6 |  |
| DC | 6 |  |
| FL | 6 |  |
| AZ | 6 |  |
| IL | 6 |  |
| SC | 5 |  |
| OR | 5 |  |
| NE | 5 |  |
| HI | 4 |  |
| UT | 4 |  |
| AR | 4 |  |
| NY | 4 |  |
| TN | 3 |  |
| KS | 3 |  |
| MO | 3 |  |
| PA | 3 |  |
| MA | 3 |  |
| LA | 3 |  |
| NJ | 3 |  |
| NH | 2 |  |
| OH | 2 |  |
| WY | 2 |  |
| ID | 2 |  |
| KY | 1 |  |
| RI | 1 |  |
| ME | 1 |  |
| MN | 1 |  |
| SD | 1 |  |
| WI | 1 |  |
| NM | 1 |  |
| AK | 1 |  |
| MS | 1 |  |
| AL | 1 |  |
| DE | 1 |  |
| OK | 1 |  |
| ND | 1 |  |
| Name: | count, | dtype: int64 |

Make

count 112634

nunique 34

unique [TOYOTA, CHEVROLET, NISSAN, FORD, TESLA, KIA, …

Name: Make, dtype: object Value Counts:

Make

TESLA 52078

NISSAN 12880

CHEVROLET 10182

FORD 5819

BMW 4680

KIA 4483

TOYOTA 4405

VOLKSWAGEN 2514

AUDI 2332

VOLVO 2288

CHRYSLER 1794

HYUNDAI 1412

JEEP 1152

RIVIAN 885

FIAT 822

PORSCHE 818

HONDA 792

MINI 632

MITSUBISHI 588

POLESTAR 558

MERCEDES-BENZ 506

SMART 273

JAGUAR 219

LINCOLN 168

CADILLAC 108

LUCID MOTORS 65

SUBARU 59

LAND ROVER 38

LEXUS 33

FISKER 20

GENESIS 18

AZURE DYNAMICS 7

TH!NK 3

BENTLEY 3

Name: count, dtype: int64

Model

count 112614

nunique 114

unique [RAV4 PRIME, VOLT, LEAF, BOLT EV, FUSION, MODE…

Name: Model, dtype: object Value Counts:

Model

|  |  |  |
| --- | --- | --- |
| MODEL 3 | 23135 | |
| MODEL Y | 17142 | |
| LEAF | 12880 | |
| MODEL S | 7377 | |
| BOLT EV | 4910  … | |
| 745LE | 2 |  |
| S-10 PICKUP | 1 |  |
| SOLTERRA | 1 |  |
| 918 | 1 |  |
| FLYING SPUR | 1 |  |
| Name: count, | Length: | 114, dtype: int64 |

---------- Electric\_Vehicle\_Type ----------

count 112634

nunique 2

unique [Plug-in Hybrid Electric Vehicle (PHEV), Batte… Name: Electric\_Vehicle\_Type, dtype: object

Value Counts:

Electric\_Vehicle\_Type

Battery Electric Vehicle (BEV) 86044 Plug-in Hybrid Electric Vehicle (PHEV) 26590 Name: count, dtype: int64

CAFV\_Eligibility

count 112634

nunique 3

unique [Clean Alternative Fuel Vehicle Eligible, Not … Name: CAFV\_Eligibility, dtype: object

Value Counts:

CAFV\_Eligibility

Clean Alternative Fuel Vehicle Eligible 58639 Eligibility unknown as battery range has not been researched 39236 Not eligible due to low battery range 14759 Name: count, dtype: int64

Vehicle\_Location

count 112610

nunique 758

unique [POINT (-81.80023 24.5545), POINT (-114.57245 …

Name: Vehicle\_Location, dtype: object Value Counts:

Vehicle\_Location

|  |  |
| --- | --- |
| POINT (-122.13158 47.67858) | 2916 |
| POINT (-122.2066 47.67887) | 2059 |
| POINT (-122.1872 47.61001) | 2001 |
| POINT (-122.31765 47.70013) | 1880 |
| POINT (-122.12096 47.55584) | 1852 |

|  |  |  |
| --- | --- | --- |
|  | … |  |
| POINT (-124.33152 48.05431) |  | 1 |
| POINT (-77.41203 39.41574) |  | 1 |
| POINT (-123.61022 46.35588) |  | 1 |
| POINT (-112.04165 40.68741) |  | 1 |
| POINT (-116.91895 47.40077) |  | 1 |

Name: count, Length: 758, dtype: int64

Electric\_Utility

count 112191

nunique 73

unique [nan, PACIFICORP, PUGET SOUND ENERGY INC, PUD …

Name: Electric\_Utility, dtype: object Value Counts:

Electric\_Utility

PUGET SOUND ENERGY INC||CITY OF TACOMA - (WA) 40247

PUGET SOUND ENERGY INC 22172

CITY OF SEATTLE - (WA)|CITY OF TACOMA - (WA) 21447

BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF CLARK COUNTY - (WA) 6522

BONNEVILLE POWER ADMINISTRATION||CITY OF TACOMA - (WA)||PENINSULA LIGHT COMPANY 5053

…

BONNEVILLE POWER ADMINISTRATION||PENINSULA LIGHT COMPANY 1

BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF ASOTIN COUNTY 1

CITY OF SEATTLE - (WA) 1

BONNEVILLE POWER ADMINISTRATION||NESPELEM VALLEY ELEC COOP, INC 1

BONNEVILLE POWER ADMINISTRATION||PUD NO 1 OF CLALLAM COUNTY|PUD NO 1 OF JEFFERSON COUNTY 1

Name: count, Length: 73, dtype: int64

[ ]:

**def** numerical\_univariate\_analysis(numerical\_data):

**for** col\_name **in** numerical\_data: print("-"\*10, col\_name, "-"\*10)

print(numerical\_data[col\_name].agg(['min', 'max', 'mean', 'median',␣

𝗌'std']))

print()

[ ]:

numerical\_univariate\_analysis(numerical\_df)

[ ]:

sns.set(style="whitegrid")*# Univariate Analysis: Distribution of Numerical*␣

𝗌*Columns*

*# Plot histograms for numerical columns*

**for** column **in** numerical\_columns: plt.figure(figsize=(15, 10))

sns.histplot(df[column], kde=**True**) plt.title(f'Distribution of **{**column**}**')

plt.tight\_layout() plt.show()

## Visual Univariate Analysis on Numerical Columns Frequency Distribution

[ ]:

## Outlier Detection

[ ]:

**def** describe\_outliers(df, column): Q1 = df[column].quantile(0.25) Q3 = df[column].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR upper\_bound = Q3 + 1.5 \* IQR

outliers = df[(df[column] < lower\_bound) | (df[column] > upper\_bound)] print(f"**\**

Column: **{**column**}**")

print(f"Number of outliers: **{**len(outliers)**}**")

print(f"Percentage of outliers: **{**len(outliers) / len(df) \* 100**:**.2f**}**%") print(f"Range of outliers: **{**outliers[column].min()**}** to **{**outliers[column].

𝗌max()**}**")

print(f"Range of non-outliers: **{**df[(df[column] >= lower\_bound) &␣

𝗌(df[column] <= upper\_bound)][column].min()**}** to **{**df[(df[column] >=␣

𝗌lower\_bound) & (df[column] <= upper\_bound)][column].max()**}**")

**for** column **in** numerical\_columns: describe\_outliers(df, column)

*# Box plots for numerical columns*

**for** column **in** numerical\_columns: plt.figure(figsize=(15, 10))

sns.boxplot(x=df[column]) plt.title(f'Box Plot of **{**column**}**')

plt.tight\_layout() plt.show()

## Visual Univariate Analysis on Categorical Variables

[ ]:

*# Plot bar charts for categorical columns*

plt.figure(figsize=(15, 10))

**for** i, column **in** enumerate(categorical\_columns[:6], 1): *# Limiting to first 6*␣

𝗌*for clarity*

plt.subplot(3, 2, i)

sns.countplot(y=df[column], order=df[column].value\_counts().index[:10]) plt.title(f'Top 10 **{**column**}**')

plt.tight\_layout() plt.show()

## Bivariate Analysis

[ ]: *# 1. Relationship between Model Year and Electric Range*

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

sns.scatterplot(x='Model\_Year', y='Electric\_Range', data=df) plt.title('Model Year vs Electric Range')

plt.show()

*# 2. Comparison of Electric Range across different Electric Vehicle Types*

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

sns.boxplot(x='Electric\_Vehicle\_Type', y='Electric\_Range', data=df) plt.title('Electric Range by Vehicle Type')

plt.xticks(rotation=45) plt.show()

*# 3. Correlation between Electric Range and Base MSRP # First, let's check if Base MSRP has non-zero values* **if** df['Base\_MSRP'].sum() > 0:

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

sns.scatterplot(x='Base\_MSRP', y='Electric\_Range', data=df) plt.title('Base MSRP vs Electric Range')

plt.show()

## else:

print("Base MSRP column contains only zero values. Skipping this analysis.")

*# 4. Distribution of Electric Vehicle Types across different States*

vehicle\_type\_by\_state = df.groupby('State')['Electric\_Vehicle\_Type'].

𝗌value\_counts().unstack()

plt.figure(figsize=(15, 8)) vehicle\_type\_by\_state.plot(kind='bar', stacked=**True**) plt.title('Distribution of Electric Vehicle Types across States') plt.xlabel('State')

plt.ylabel('Count')

plt.legend(title='Electric Vehicle Type', bbox\_to\_anchor=(1.05, 1), loc='upper␣

𝗌left')

plt.tight\_layout() plt.show()

[ ]:

**import matplotlib.pyplot as plt import pandas as pd**

**import numpy as np import seaborn as sns**

df = pd.read\_csv("dataset.csv")

*# 5. Correlation matrix for numerical variables*

plt.figure(figsize=(10, 8)) correlation\_matrix = df.corr()

sns.heatmap(correlation\_matrix, annot=**True**, cmap='coolwarm', fmt='.2f') plt.title('Correlation Matrix of Numerical Features')

plt.show()

*# 6. Distribution of Electric Vehicle Types by Make*

plt.figure(figsize=(14, 7))

sns.countplot(y='Make', hue='Electric\_Vehicle\_Type', data=df, order=df['Make'].

𝗌value\_counts().index)

plt.title('Distribution of Electric Vehicle Types by Make') plt.xlabel('Count')

plt.ylabel('Make') plt.legend(title='Electric Vehicle Type') plt.show()

[ ]:

*# Assuming 'df' is your DataFrame*

df.boxplot(by="CAFV\_Eligibility", column=['Electric\_Range'])

*# Rotate x-axis labels by 90 degrees*

plt.xticks(rotation=90)

*# Show the plot*

plt.show()

[ ]:

[ ]:

[ ]:

ev\_count\_by\_state = df.groupby('State').size().

𝗌reset\_index(name='Number\_of\_EV\_Vehicles')

ev\_count\_by\_state

# Task 2: Create a Choropleth using plotly.express to display the number of EV vehicles based on location

! pip install plotly

**import plotly.express as px**

[ ]:

*# Count the number of EVs per state*

ev\_count\_by\_state = df['State'].value\_counts().reset\_index() ev\_count\_by\_state.columns = ['State', 'EV\_Count']

*# Create the Choropleth map*

fig = px.choropleth(ev\_count\_by\_state,

locations='State', locationmode="USA-states", color='EV\_Count', scope="usa",

color\_continuous\_scale="Viridis", title="Number of Electric Vehicles by State")

*# Update the layout* fig.update\_layout( title\_x=0.5,

geo\_scope='usa',

)

fig.show()

*# Save the plot as an HTML file*

fig.write\_html("ev\_choropleth\_map.html")

print("Choropleth map has been created and saved as 'ev\_choropleth\_map.html'.") print("**\**

Top 5 states by EV count:") print(ev\_count\_by\_state.head().to\_string(index=**False**))

[ ]:

**import pandas as pd**

**import plotly.express as px**

*# Load the dataset*

df = pd.read\_csv('dataset.csv', encoding='ascii')

*# Count the number of EVs per postal code*

ev\_count\_by\_postal = df['Postal Code'].value\_counts().reset\_index() ev\_count\_by\_postal.columns = ['Postal Code', 'EV\_Count']

*# Merge the count with the original dataframe to get location data*

df\_merged = df.merge(ev\_count\_by\_postal, on='Postal Code')

*# Extract latitude and longitude from the 'Vehicle Location' column*

df\_merged['Longitude'] = df\_merged['Vehicle Location'].str.extract('POINT␣

𝗌\(([-\d.]+) ')

df\_merged['Latitude'] = df\_merged['Vehicle Location'].str.extract(' ([-\d.

𝗌]+)\)')

*# Convert to numeric*

df\_merged['Longitude'] = pd.to\_numeric(df\_merged['Longitude']) df\_merged['Latitude'] = pd.to\_numeric(df\_merged['Latitude'])

*# Create the scatter plot on a map*

fig = px.scatter\_mapbox(df\_merged,

lat='Latitude', lon='Longitude', color='EV\_Count', size='EV\_Count', hover\_name='Postal Code',

hover\_data=['City', 'State', 'EV\_Count'], color\_continuous\_scale="Viridis", size\_max=15,

zoom=3,

title="Number of Electric Vehicles by Postal Code")

fig.update\_layout(mapbox\_style="open-street-map") fig.update\_layout(margin={"r":0,"t":0,"l":0,"b":0})

*# Save the plot as an HTML file*

fig.write\_html("ev\_postal\_code\_map.html") fig.show()

print("Scatter map based on postal codes has been created and saved as␣

𝗌'ev\_postal\_code\_map.html'.")

print("**\**

Top 10 postal codes by EV count:") print(ev\_count\_by\_postal.head(10).to\_string(index=**False**))

*# Display some statistics*

print("**\**

Total number of unique postal codes:", len(ev\_count\_by\_postal)) print("Average number of EVs per postal code:",␣

𝗌round(ev\_count\_by\_postal['EV\_Count'].mean(), 2))

print("Median number of EVs per postal code:", ev\_count\_by\_postal['EV\_Count'].

𝗌median())

print("Maximum number of EVs in a single postal code:",␣

𝗌ev\_count\_by\_postal['EV\_Count'].max())

[ ]:

[ ]:

# Task 3: Create a Racing Bar Plot to display the animation of EV Make and its count each year.

!pip install bar-chart-race

[ ]:

**import bar\_chart\_race as bcr import warnings**

*# Convert 'Model Year' to string for grouping*

df['Model Year'] = df['Model Year'].astype(str)

*# Group the data by 'Model Year' and 'Make', then count the occurrences*

grouped\_data = df.groupby(['Model Year', 'Make']).size().

𝗌reset\_index(name='Count')

*# Pivot the data to have 'Model Year' as the index and 'Make' as columns*

pivoted\_data = grouped\_data.pivot(index='Model Year', columns='Make',␣

𝗌values='Count')

*# Fill missing values with 0 (for years where some makes might have no entries)*

pivoted\_data = pivoted\_data.fillna(0)

*# Create the bar chart race animation and save it as a GIF*

bcr.bar\_chart\_race(df=pivoted\_data, filename='EV\_racing\_bar\_plot.gif', orientation='h', sort='desc', n\_bars=10,

title='EV Make Count Over the Years',␣

𝗌filter\_column\_colors=**True**, period\_length=1000)

[ ]: