Electric Vehicle (EV) Market Segmentation Report

NEERAJA RANI MAJJI

GITHUB:-https://github.com/neerajamajji23/Ev-Market-Segmentation-Feynn-Labs-

1. Introduction

The Electric Vehicle (EV) market is evolving rapidly, fueled by heightened environmental awareness, advancements in technology, and supportive governmental policies. This transformation represents a shift in consumer preferences towards sustainable transportation options. The following report delves into customer demographics, vehicle preferences, and market trends to provide actionable insights for businesses looking to capitalize on this growing market.

- **Urban and Suburban Dominance**: Urban areas have the highest EV adoption rates due to better infrastructure and awareness, while suburban regions exhibit significant growth potential.
- **Vehicle Preferences**: Sedans and SUVs are the most sought-after categories among high-income groups, while two-wheelers are popular with younger and budget-conscious consumers.
- **Income and Loan Dynamics**: Higher income levels correlate with a preference for premium EV types, whereas middle-income groups are influenced by the availability of flexible loan options.

2. Explained Process (Framework)

2.1 Data Collection

Two primary datasets were utilized for this analysis:

- **Customer Demographics**: Included data on age, gender, income levels, geographical locations, EV preferences, loan statuses, and education levels.
- **Market Trends**: Focused on EV sales data, vehicle types, regional adoption rates, and target consumer groups.

2.2 Data Preprocessing

- **Handling Missing Data**: Missing values in both datasets were imputed using statistical techniques to ensure completeness.

- **Standardization**: Continuous variables, such as income and age, were standardized to improve model performance.
- **Encoding Categorical Variables**: Variables like gender, geography, and education levels were converted into numerical formats for machine learning algorithms.

2.3 Analytical Techniques

- **Exploratory Data Analysis (EDA)**: Used visualizations like bar charts, heatmaps, and scatter plots to identify patterns and trends.
- **Clustering**: KMeans clustering segmented the market into distinct consumer groups based on their preferences and behaviors.
- **Principal Component Analysis (PCA)**: Reduced the dimensionality of data, enabling efficient visualization and analysis.

3. Key Insights

- **Demographic Trends**:
  - Age: Younger individuals (18–35) prefer affordable two-wheelers.
  - Income: Higher-income groups favor premium EVs such as sedans and SUVs.
- **Geographical Insights**:
  - Urban: High adoption rates due to better charging infrastructure and eco-conscious populations.
  - Suburban: Emerging as a promising market with growing interest in EVs.
- **Behavioral Insights**:
  - Loan Influence: Loan accessibility significantly impacts purchasing decisions, especially in the middle-income segment.

4. Solutions and Recommendations

1. **Target Younger Consumers**:
   a. Develop budget-friendly two-wheeler models with improved range and affordability.
2. **Focus on Suburban Markets**:
   a. Enhance marketing campaigns to highlight the practicality and long-term savings of EVs in suburban regions.
3. **Collaborate with Financial Institutions**:

    a. Introduce innovative loan schemes to attract middle-income buyers.
4. **Expand Charging Infrastructure**:
    a. Partner with local governments to develop charging networks in suburban and rural areas.
5. **Product Customization**:
    a. Offer customizable options such as battery upgrades and smart features to appeal to tech-savvy consumers.

5. Detailed Explanation of Graphs

5.1 Correlation Heatmaps

Heatmaps illustrate relationships between variables:

- **Age vs. Income**: Weak negative correlation suggests diverse EV preferences across different age groups.
- **Income vs. Vehicle Type**: Stronger preferences for premium vehicles in higher-income brackets.

5.2 Geographical Distribution

Maps showcase regional EV adoption trends:

- Urban areas dominate EV sales due to infrastructure and environmental awareness.
- Suburban regions show potential for growth with strategic investments in infrastructure.

5.3 Vehicle Preferences

Bar charts highlight:

- Sedans and SUVs leading in sales among high-income segments.
- Two-wheelers gaining popularity among younger, cost-conscious consumers.

5.4 Clustering Results

Cluster plots visualize market segmentation:

- Segments are defined by income levels, geography, and vehicle type preferences, aiding targeted marketing strategies.

6. Representation

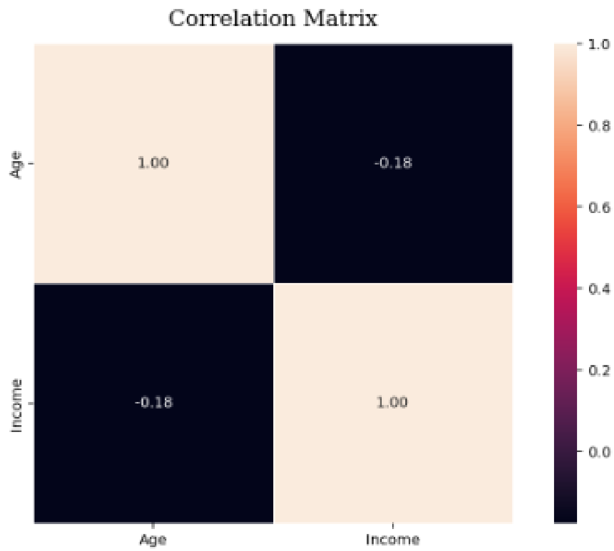
6.1 Graphical Insights

- Correlation heatmaps demonstrate variable relationships.
- Bar charts illustrate sales distribution across income groups and vehicle types.
- Cluster plots provide a visual summary of market segmentation.

In [18]:

```
corr_df1 = df1[["Age", "Income"]]
plt.figure(figsize=(10, 6))
sns.heatmap(data=corr_df1.corr(), annot=True, square=True, fmt='.2f', linewidths=.3)
plt.title('Correlation Matrix', family='serif', size=15, pad=12);
```

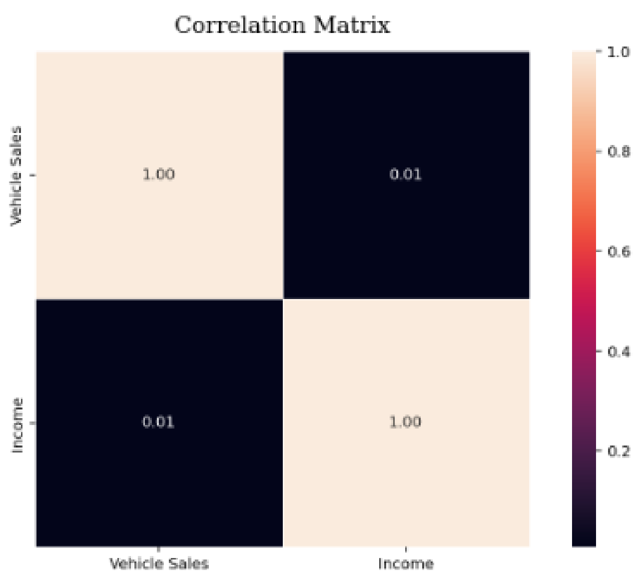
Out[18]:



In [23]:

```
corr_df2 = df2[["Vehicle Sales","Income"]]
plt.figure(figsize=(10, 6))
sns.heatmap(data=corr_df2.corr(), annot=True, square=True, fmt='.2f', linewidths=.3)
plt.title('Correlation Matrix', family='serif', size=15, pad=12);
```

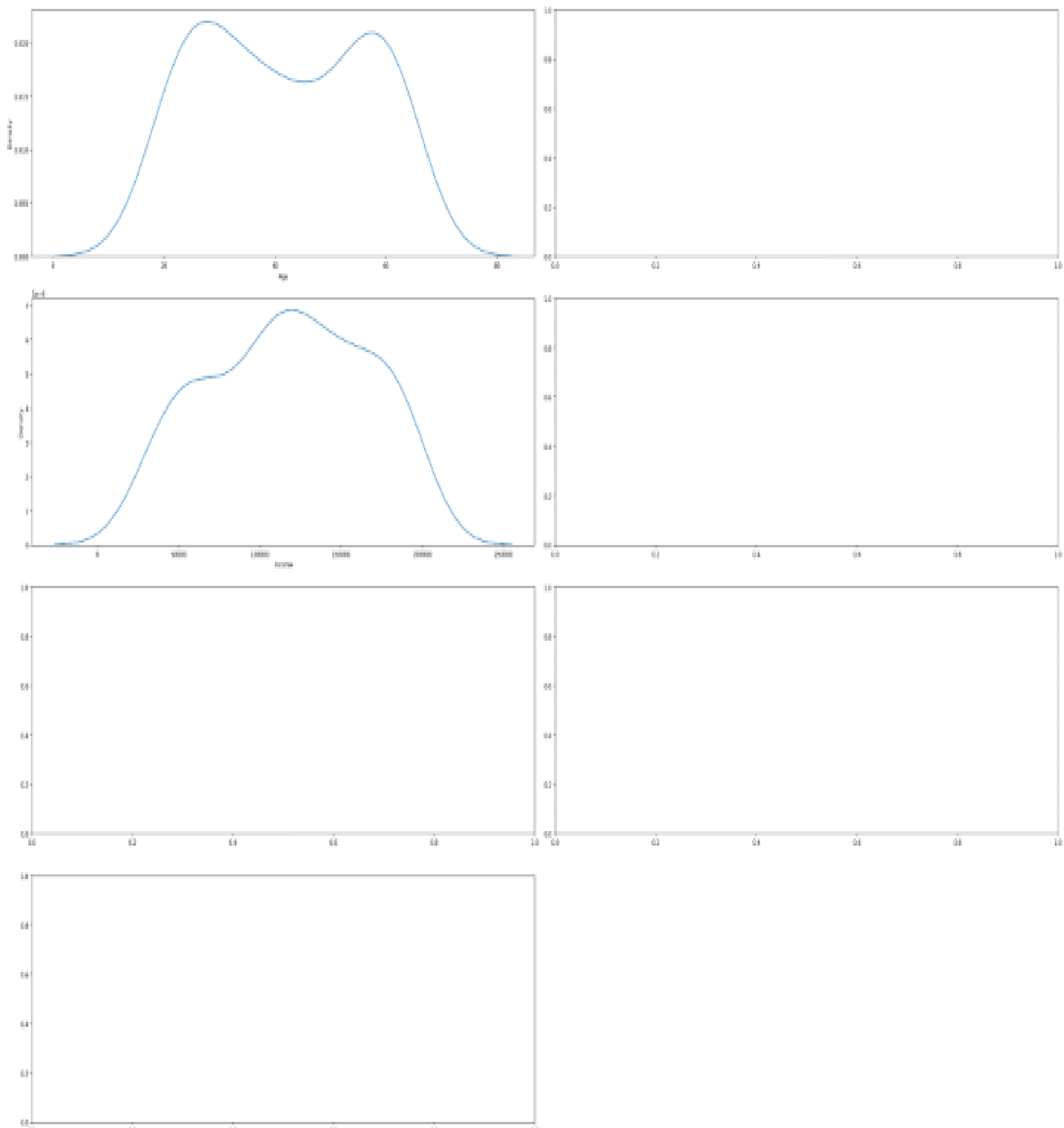
Out[23]:



In [20]:

```python
fig, axes = plt.subplots(4, 2, figsize=(18, 18))
axes = axes.flatten()
for i, col in enumerate(df1.columns):
    if df1[col].dtype != 'object':
        axes[i].set_xlabel(col)
        sns.kdeplot(df1[col], ax=axes[i])

for j in range(len(df1.columns), len(axes)):
    fig.delaxes(axes[j])

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

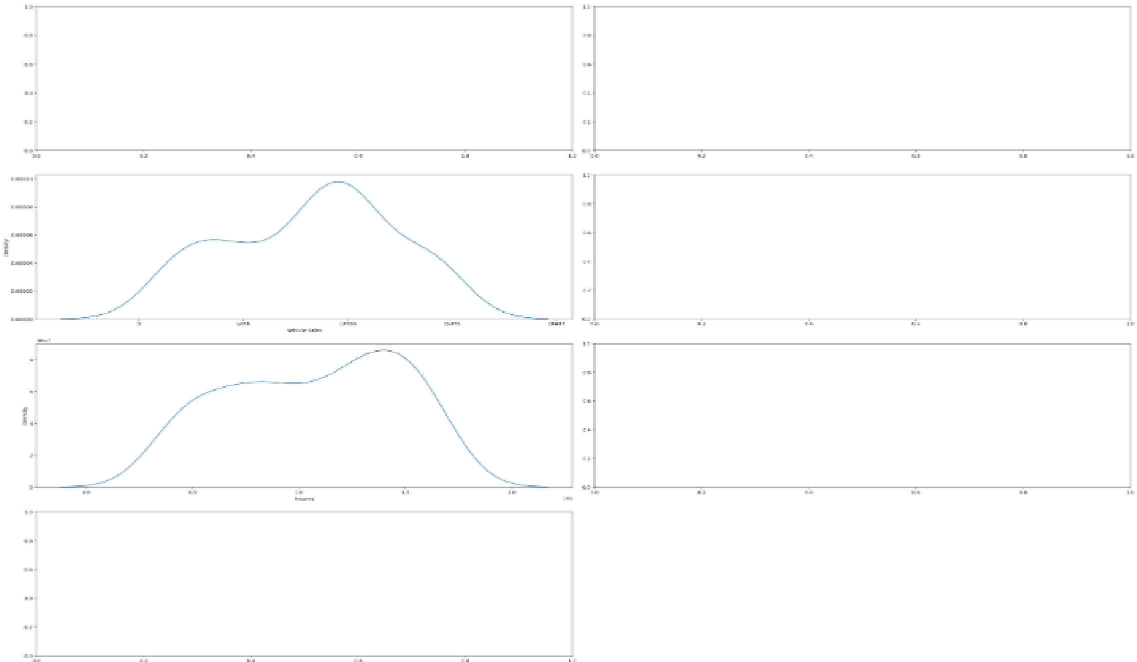
Out[20]:



In [19]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df2=pd.read_csv("ev market.csv")

fig, axes = plt.subplots(5, 2, figsize=(16, 25))
axes = axes.flatten()

for i, col in enumerate(df2.columns):
    if df2[col].dtype != "object":
        axes[i].set_xlabel(col)
        sns.kdeplot(df2[col], ax=axes[i])

for j in range(len(df2.columns), len(axes)):
    fig.delaxes(axes[j])

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

Out[19]:

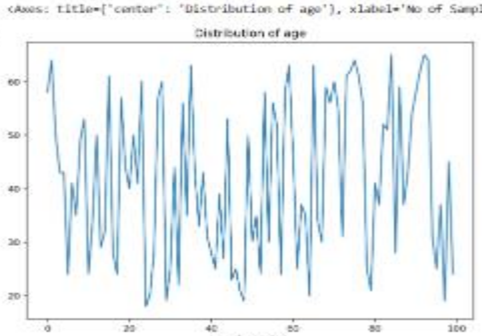


In [28]:

```python
df1 = pd.read_csv("ev data.csv")
df1['Age'].plot(xlabel='No of Samples', y='Age', title='Distribution of age')
```

Out[28]: <Axes: title={'center': 'Distribution of age'}, xlabel='No of Samples'>

Out[28]:

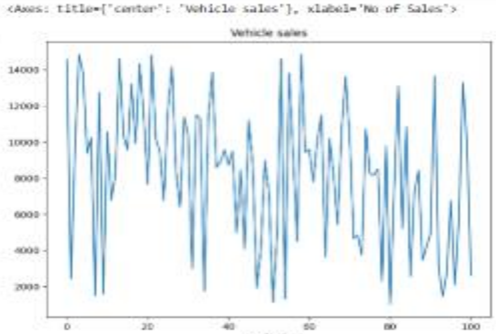


In [24]:

```python
df2 = pd.read_csv("ev market.csv")
df2['Vehicle Sales'].plot(xlabel='No of Sales', title='Vehicle sales')
```

Out[24]: <Axes: title={'center': 'Vehicle sales'}, xlabel='No of Sales'>

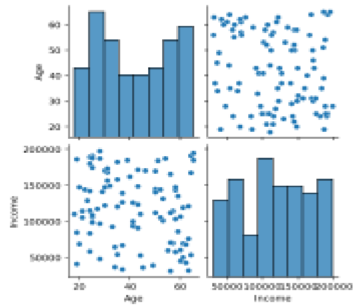
Out[24]:



In [25]:
```
sns.pairplot(df1)
```

Out[25]: <seaborn.axisgrid.PairGrid at 0x7f6c73736290>

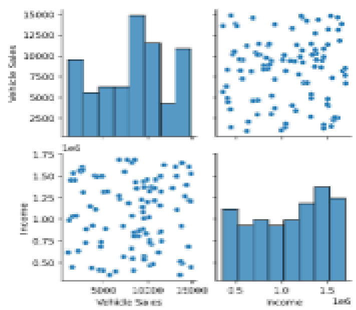
Out[25]:



In [26]:
```
sns.pairplot(df2)
```

Out[26]: <seaborn.axisgrid.PairGrid at 0x7f6c7358b100>

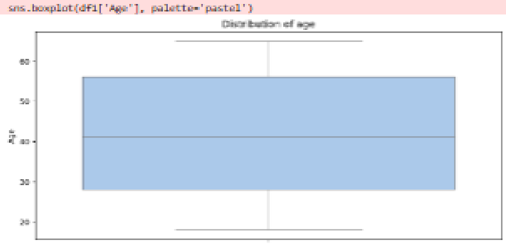
Out[26]:



In [6]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df1 = pd.read_csv("ev data.csv")
plt.figure(figsize=(10, 5))
sns.boxplot(df1['Age'], palette='pastel')
plt.title('Distribution of age')
plt.show()
```

```
/tmp/ipykernel_381/1405335141.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(df1['Age'], palette='pastel')
```

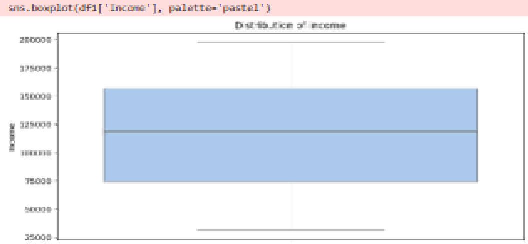
Out[6]:



In [7]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df2 = pd.read_csv("ev market.csv")
plt.figure(figsize=(10, 5))
sns.boxplot(df1['Income'], palette='pastel')
plt.title('Distribution of income')
plt.show()
```

```
/tmp/ipykernel_381/498913726.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(df1['Income'], palette='pastel')
```

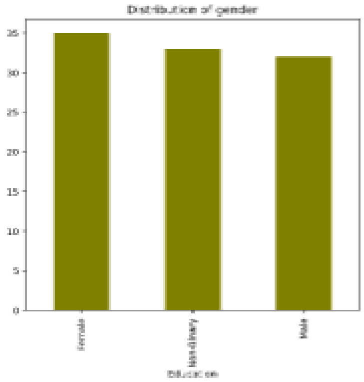
Out[7]:



In [14]:
```
plt.figure(figsize=(6,6))
df1['Gender'].value_counts().plot(kind='bar',color='olive')
plt.title('Distribution of gender')
plt.xlabel('Education')
plt.show
```

Out[14]: <function matplotlib.pyplot.show(close=None, block=None)>

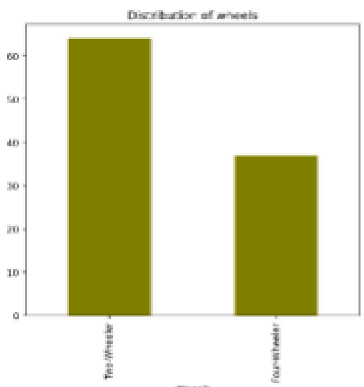
Out[14]:



In [16]:
```
plt.figure(figsize=(6,6))
df2['Wheels'].value_counts().plot(kind='bar',color='olive')
plt.title('Distribution of wheels')
plt.show
```

Out[16]: <function matplotlib.pyplot.show(close=None, block=None)>

Out[16]:



6.2 Data Tables

Data tables refer to a structured format where the model's predictions or results are presented in a tabular form, with columns representing different features or variables and rows representing individual data points, essentially providing a clear and organized way to view the model's outcomes for each data sample within a dataset.

[2]:

```
df1=pd.read_csv("ev data.csv")
df1.head()
```

t[2]:

	Age	Gender	Income	Geography	Preferred_EV_Type	Loan	Education
0	58	Male	57864	Rural	Sedan	Car Loan	Diploma
1	64	Male	53347	Suburban	Sedan	Car Loan	Master's
2	50	Non-Binary	144759	Rural	Hatchback	Car Loan	Doctorate
3	43	Female	123791	Suburban	Truck	No Loan	Master's
4	43	Male	123432	Suburban	SUV	Car Loan	High School

ı [6]:

```
df2=pd.read_csv("ev market.csv")
df2.head()
```

ıt[6]:

	Company Name	Vehicle Type	Vehicle Sales	Ages (Between)	Income	Geographical Location	Wheels
0	Tata Motors	Electric Car	14564	18-35	687646	Hubli	Four-Wheeler
1	Ola Electric	Electric Auto	2429	18-35	1552750	Kozhikode	Two-Wheeler
2	Mahindra Electric	Electric Car	9568	25-45	1010401	Pune	Four-Wheeler
3	Ather Energy	Scooter	14845	30-50	1530766	Coimbatore	Two-Wheeler
4	Hero Electric	Scooter	13796	20-40	922869	Bangalore	Two-Wheeler

ı [7]:

```
import pandas as pd
df1=pd.read_csv("ev data.csv")
df1.corr(numeric_only=True)
```

ıt[7]:

	Age	Income
Age	1.000000	-0.175993
Income	-0.175993	1.000000

ı [8]:

```
import pandas as pd
df2=pd.read_csv("ev market.csv")
df2.corr(numeric_only=True)
```

ıt[8]:

	Vehicle Sales	Income
Vehicle Sales	1.000000	0.008137
Income	0.008137	1.000000

n [6]:
```
print(' DATASET 1:')
print(df1.info())
print(' DATASET 2:')
print(df2.info())
```

```
 DATASET 1:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Age                100 non-null    int64 
 1   Gender             100 non-null    object
 2   Income             100 non-null    int64 
 3   Geography          100 non-null    object
 4   Preferred_EV_Type  100 non-null    object
 5   Loan               100 non-null    object
 6   Education          100 non-null    object
dtypes: int64(2), object(5)
memory usage: 5.6+ KB
None
 DATASET 2:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101 entries, 0 to 100
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   Company Name           101 non-null    object
 1   Vehicle Type           101 non-null    object
 2   Vehicle Sales          101 non-null    int64 
 3   Ages (Between)         101 non-null    object
 4   Income                 101 non-null    int64 
 5   Geographical Location  101 non-null    object
 6   Wheels                 101 non-null    object
dtypes: int64(2), object(5)
memory usage: 5.6+ KB
None
```

In [26]:
```
df1.columns.values.tolist()
```

Out[26]:
```
['Age',
 'Gender',
 'Income',
 'Geography',
 'Preferred_EV_Type',
 'Loan',
 'Education']
```

In [27]:
```
df2.columns.values.tolist()
```

Out[27]:
```
['Company Name',
 'Vehicle Type',
 'Vehicle Sales',
 'Ages (Between)',
 'Income',
 'Geographical Location',
 'Wheels']
```

In [11]:

```
d1 = df1.describe()
d2 = df2.describe()
display( d1,d2)
```

Out[11]:

	Age	Income
count	100.000000	100.000000
mean	41.690000	117458.950000
std	14.818563	48176.533386
min	18.000000	31277.000000
25%	28.000000	74386.750000
50%	41.000000	118486.000000
75%	56.000000	156788.500000
max	65.000000	197533.000000

Out[11]:

	Vehicle Sales	Income
count	101.000000	1.010000e+02
mean	8260.267327	1.070446e+06
std	3972.953896	4.000992e+05
min	1005.000000	3.572080e+05
25%	4850.000000	7.365100e+05
50%	8735.000000	1.115476e+06
75%	10843.000000	1.430262e+06
max	14861.000000	1.693175e+06

In [22]:

```python
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

# Assuming df1 and df2 have already been defined
df1 = df1.dropna(subset=['Age', 'Income'])
df2 = df2.dropna(subset=['Vehicle Sales'])

# Align df2 with df1's index
df2 = df2.loc[df1.index]

X = df1[['Age', 'Income']]
y = df2['Vehicle Sales']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
y_pred = dt_model.predict(X_test)

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nAccuracy Score:", accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
 [[0 0 0 ... 0 0 1]
 [0 0 0 ... 0 0 1]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]

Classification Report:
              precision    recall  f1-score   support

        1005       0.00      0.00      0.00       1.0
        1583       0.00      0.00      0.00       1.0
        2115       0.00      0.00      0.00       1.0
        2429       0.00      0.00      0.00       0.0
        2572       0.00      0.00      0.00       0.0
        2723       0.00      0.00      0.00       0.0
        2973       0.00      0.00      0.00       1.0
        3462       0.00      0.00      0.00       1.0
        3756       0.00      0.00      0.00       1.0
        4085       0.00      0.00      0.00       1.0
        4213       0.00      0.00      0.00       0.0
        4489       0.00      0.00      0.00       0.0
        4663       0.00      0.00      0.00       0.0
        4850       0.00      0.00      0.00       1.0
        4948       0.00      0.00      0.00       1.0
        4982       0.00      0.00      0.00       1.0
        5215       0.00      0.00      0.00       1.0
        7200       0.00      0.00      0.00       0.0
        7391       0.00      0.00      0.00       0.0
        7675       0.00      0.00      0.00       0.0
        7882       0.00      0.00      0.00       0.0
        8165       0.00      0.00      0.00       1.0
        8211       0.00      0.00      0.00       1.0
        8364       0.00      0.00      0.00       0.0
        8509       0.00      0.00      0.00       1.0
        8724       0.00      0.00      0.00       1.0
        9261       0.00      0.00      0.00       0.0
        9442       0.00      0.00      0.00       0.0
        9567       0.00      0.00      0.00       1.0
        9578       0.00      0.00      0.00       1.0
       10180       0.00      0.00      0.00       1.0
       10195       0.00      0.00      0.00       1.0
       10246       0.00      0.00      0.00       0.0
       10397       0.00      0.00      0.00       1.0
       10561       0.00      0.00      0.00       1.0
       10733       0.00      0.00      0.00       0.0
       11222       0.00      0.00      0.00       1.0
       11291       0.00      0.00      0.00       1.0
       11512       0.00      0.00      0.00       0.0
       13224       0.00      0.00      0.00       0.0
       13625       0.00      0.00      0.00       1.0
       13796       0.00      0.00      0.00       1.0
       13814       0.00      0.00      0.00       1.0
       14189       0.00      0.00      0.00       1.0
       14329       0.00      0.00      0.00       1.0
       14564       0.00      0.00      0.00       1.0
       14603       0.00      0.00      0.00       1.0
       14812       0.00      0.00      0.00       0.0
       14845       0.00      0.00      0.00       0.0
       14861       0.00      0.00      0.00       0.0

    accuracy                           0.00      30.0
   macro avg       0.00      0.00      0.00      30.0
weighted avg       0.00      0.00      0.00      30.0


Accuracy Score: 0.0
```

In [24]:

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split  # missing import

# Feature scaling (important for KNN)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Splitting the data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

# Training the K-Nearest Neighbors model
knn_model = KNeighborsClassifier(n_neighbors=5)  # Adjust 'n_neighbors' if needed
knn_model.fit(X_train, y_train)

# Making predictions
y_pred = knn_model.predict(X_test)

# Evaluating the model
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nAccuracy Score:", accuracy_score(y_test, y_pred))  # Added closing parenthesis
```

```
Classification Report:
              precision    recall  f1-score   support

        1005       0.00      0.00      0.00       1.0
        1133       0.00      0.00      0.00       0.0
        1313       0.00      0.00      0.00       0.0
        1471       0.00      0.00      0.00       0.0
        1488       0.00      0.00      0.00       0.0
        1583       0.00      0.00      0.00       1.0
        1751       0.00      0.00      0.00       0.0
        1908       0.00      0.00      0.00       0.0
        2115       0.00      0.00      0.00       1.0
        2285       0.00      0.00      0.00       0.0
        2429       0.00      0.00      0.00       0.0
        2973       0.00      0.00      0.00       1.0
        3462       0.00      0.00      0.00       1.0
        3756       0.00      0.00      0.00       1.0
        4085       0.00      0.00      0.00       1.0
        4213       0.00      0.00      0.00       0.0
        4663       0.00      0.00      0.00       0.0
        4850       0.00      0.00      0.00       1.0
        4948       0.00      0.00      0.00       1.0
        4982       0.00      0.00      0.00       1.0
        5215       0.00      0.00      0.00       1.0
        6751       0.00      0.00      0.00       0.0
        7200       0.00      0.00      0.00       0.0
        8165       0.00      0.00      0.00       1.0
        8211       0.00      0.00      0.00       1.0
        8364       0.00      0.00      0.00       0.0
        8426       0.00      0.00      0.00       0.0
        8509       0.00      0.00      0.00       1.0
        8724       0.00      0.00      0.00       1.0
        9567       0.00      0.00      0.00       1.0
        9578       0.00      0.00      0.00       1.0
       10180       0.00      0.00      0.00       1.0
       10195       0.00      0.00      0.00       1.0
       10397       0.00      0.00      0.00       1.0
       10561       0.00      0.00      0.00       1.0
       11222       0.00      0.00      0.00       1.0
       11291       0.00      0.00      0.00       1.0
       13625       0.00      0.00      0.00       1.0
       13796       0.00      0.00      0.00       1.0
       13814       0.00      0.00      0.00       1.0
       14189       0.00      0.00      0.00       1.0
       14329       0.00      0.00      0.00       1.0
       14564       0.00      0.00      0.00       1.0
       14603       0.00      0.00      0.00       1.0

    accuracy                           0.00      30.0
   macro avg       0.00      0.00      0.00      30.0
weighted avg       0.00      0.00      0.00      30.0


Accuracy Score: 0.0
```

8. Conclusion

The EV market is poised for substantial growth, driven by increasing eco-consciousness, technological innovations, and favorable policies. By leveraging insights from this report, companies can align their strategies to address consumer needs effectively and expand their presence in key market segments.