**NAME: AISHANI DATTA**

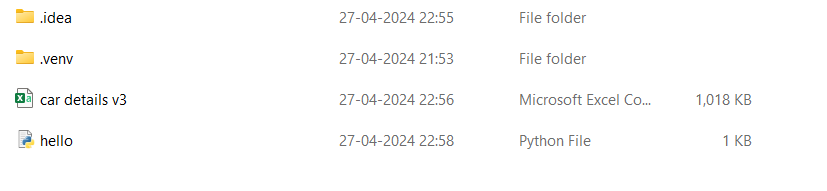
**Project 2**

**Problem Statement 1: EV Market in India using Segmentation Analysis**

**EV MARKET IN INDIA (Explained with visualization charts)**

1. **Moving 1st csv file from source path to destination path**

# src\_pth = r"E:\INTERNSHIPS\Feynn Labs Internships\Project 2\car details v3.csv"  
# dest\_pth = os.path.join(os.getcwd(), 'car details v3.csv')  
# shutil.copy(src\_pth, dest\_pth)

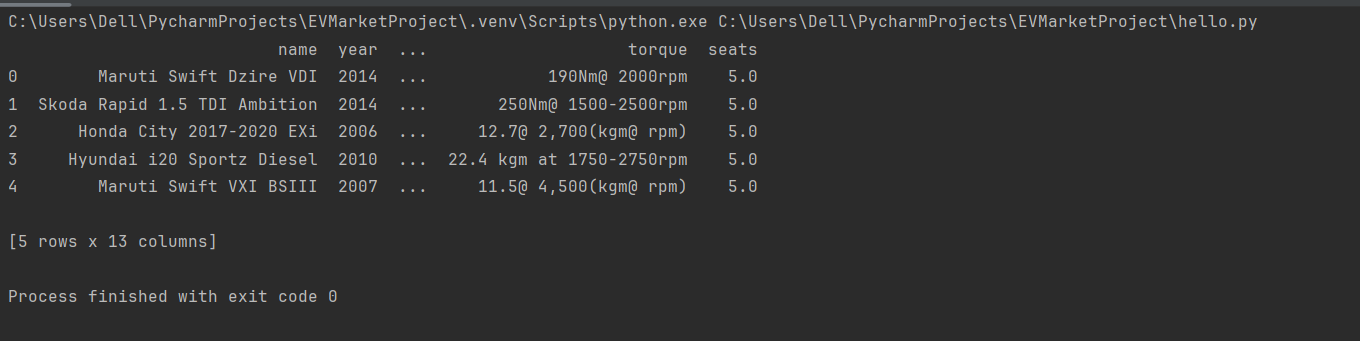


1. **Fetching Datasets**

# fetching dataset - 1

df1 = pd.read\_csv('car details v3.csv')

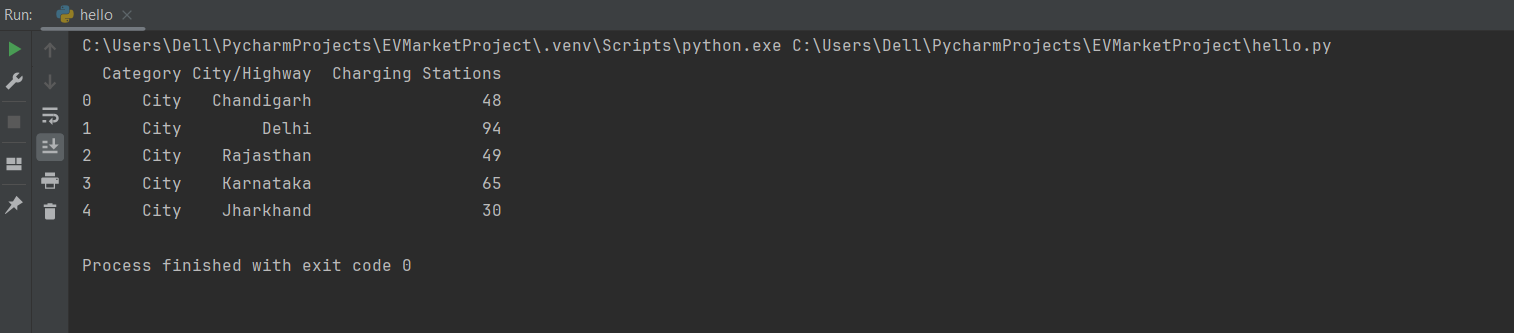
print(df1.head())



# fetching dataset - 2

df2 = pd.read\_csv('City-wise EV charging stations available in India.csv')

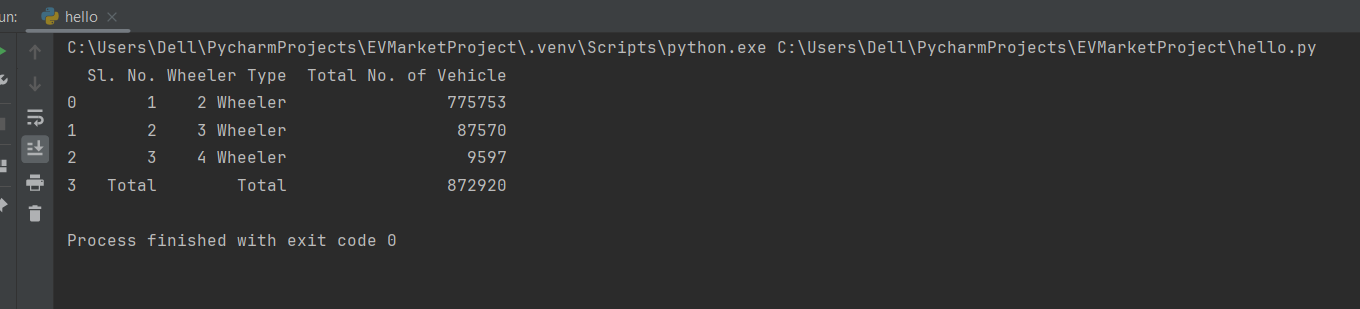
print(df2.head())



# fetching dataset - 3

df3 = pd.read\_csv('EV category-wise distribution sales to Consumers as per 2-8-23.csv')

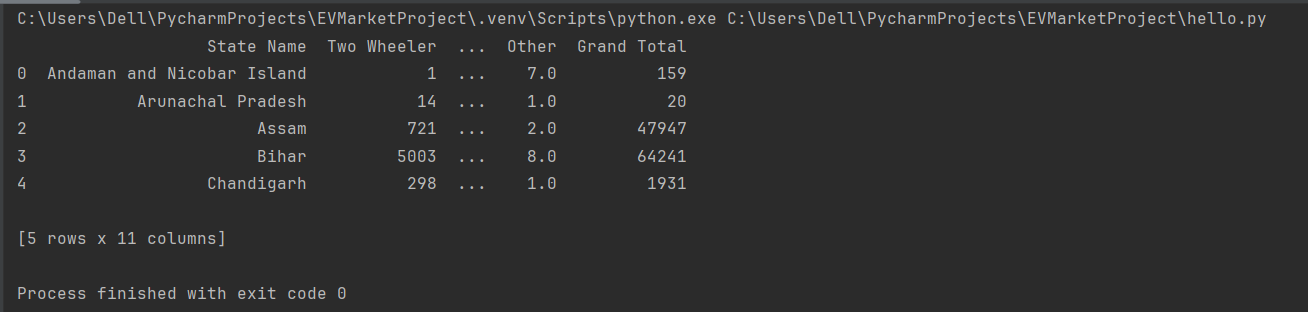
print(df3.head())



# fetching dataset – 4

df4 = pd.read\_csv('State-wise current sales of EV vehicles in various segments.csv')

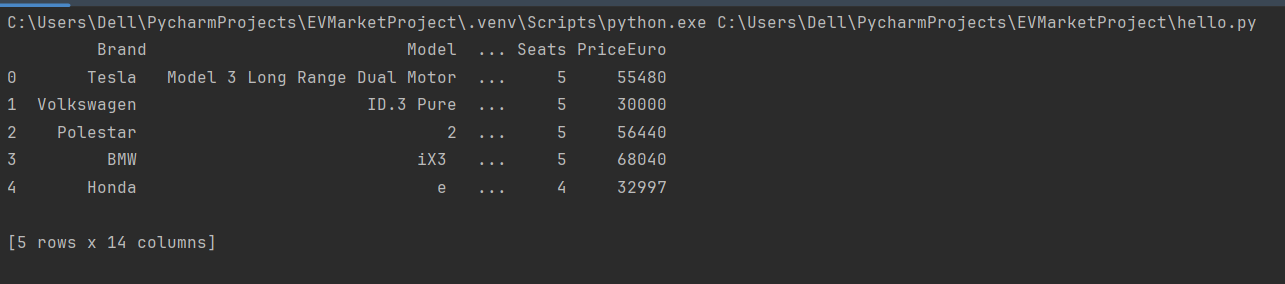
print(df4.head())



# fetching dataset – 5

Df5 = pd.read\_csv(' EV vehicles based on brands’.csv)

print(df5.head())



1. **Exploratory Data Analysis (EDA)**

**C.1 Analysing the datasets**

# checking the shape (# of rows and columns) of the datasets

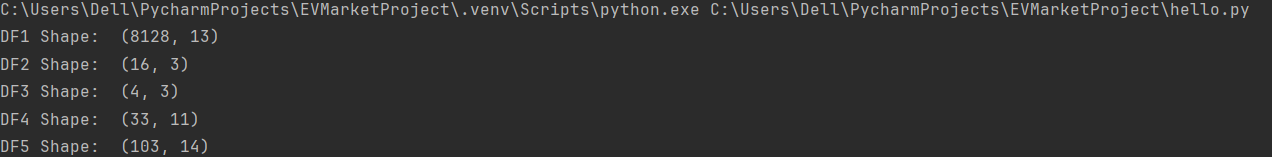
print('DF1 Shape: ', df1.shape)

print('DF2 Shape: ', df2.shape)

print('DF3 Shape: ', df3.shape)

print('DF4 Shape: ', df4.shape)

print('DF5 Shape: ', df5.shape)



# checking the info (columns, datatypes, nulls) of the datasets

print(' <<< DATASET 1 -----------------------------------------------------------')

print(df1.info())

print(' <<< DATASET 2 -----------------------------------------------------------')

print(df2.info())

print(' <<< DATASET 3 -----------------------------------------------------------')

print(df3.info())

print(' <<< DATASET 4 -----------------------------------------------------------')

print(df4.info())

print(' <<< DATASET 5-----------------------------------------------------------')

print(df5.info())

<<< DATASET 1 -----------------------------------------------------------

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 8128 entries, 0 to 8127

Data columns (total 13 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 name 8128 non-null object

1 year 8128 non-null int64

2 selling\_price 8128 non-null int64

3 km\_driven 8128 non-null int64

4 fuel 8128 non-null object

5 seller\_type 8128 non-null object

6 transmission 8128 non-null object

7 owner 8128 non-null object

8 mileage 7907 non-null object

9 engine 7907 non-null object

10 max\_power 7913 non-null object

11 torque 7906 non-null object

12 seats 7907 non-null float64

dtypes: float64(1), int64(3), object(9)

memory usage: 825.6+ KB

None

<<< DATASET 2 -----------------------------------------------------------

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 16 entries, 0 to 15

Data columns (total 3 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Category 16 non-null object

1 City/Highway 16 non-null object

2 Charging Stations 16 non-null int64

dtypes: int64(1), object(2)

memory usage: 512.0+ bytes

None

<<< DATASET 3 -----------------------------------------------------------

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 4 entries, 0 to 3

Data columns (total 3 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Sl. No. 4 non-null object

1 Wheeler Type 4 non-null object

2 Total No. of Vehicle 4 non-null int64

dtypes: int64(1), object(2)

memory usage: 224.0+ bytes

None

<<< DATASET 4 -----------------------------------------------------------

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 33 entries, 0 to 32

Data columns (total 11 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 State Name 33 non-null object

1 Two Wheeler 33 non-null int64

2 Three Wheeler 29 non-null float64

3 Four Wheeler 33 non-null int64

4 Goods Vehicles 29 non-null float64

5 Public Service Vehicle 25 non-null float64

6 Special Category Vehicles 10 non-null float64

7 Ambulance/Hearses 6 non-null float64

8 Construction Equipment Vehicle 6 non-null float64

9 Other 29 non-null float64

10 Grand Total 33 non-null int64

[8 rows x 10 columns] <<< DATASET 5 >>> Seats PriceEuro

count 103.000000 103.000000

mean 4.883495 55811.563107

std 0.795834 34134.665280

min 2.000000 20129.000000

25% 5.000000 34429.500000

50% 5.000000 45000.000000

75% 5.000000 65000.000000

max 7.000000 215000.000000

Process finished with exit code 0

# getting a statistical summary of the datasets

df1 = df1.describe()

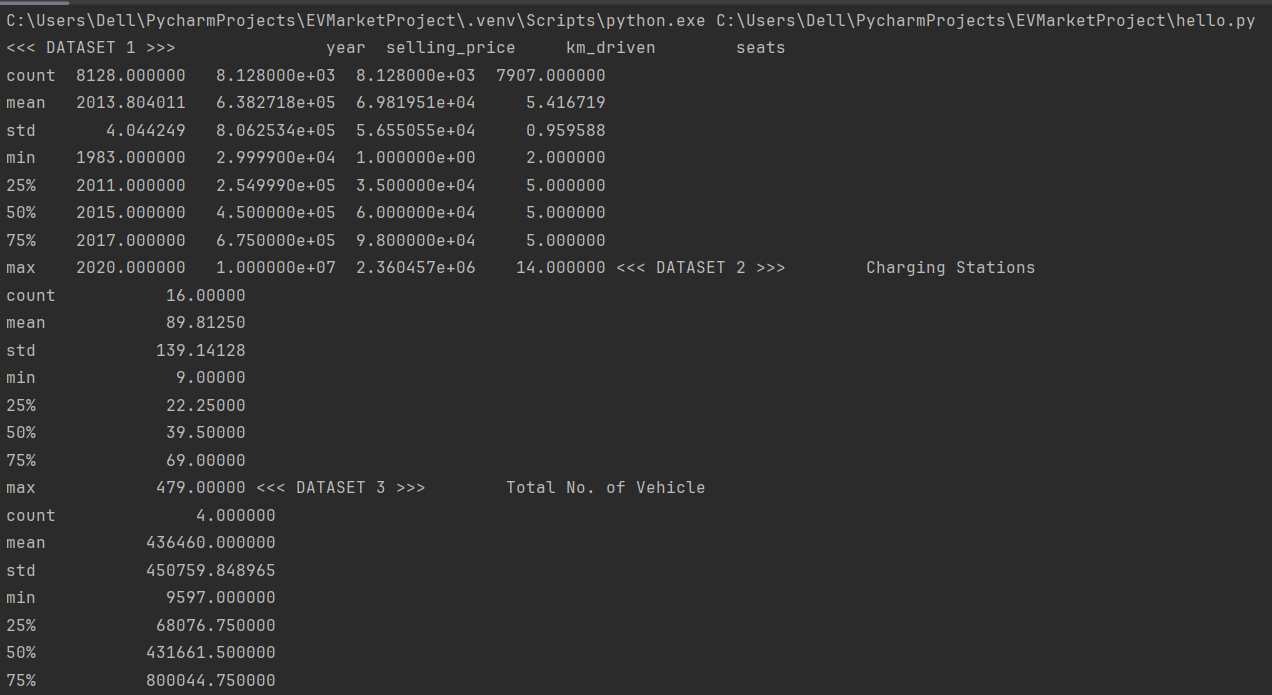
df2 = df2.describe()

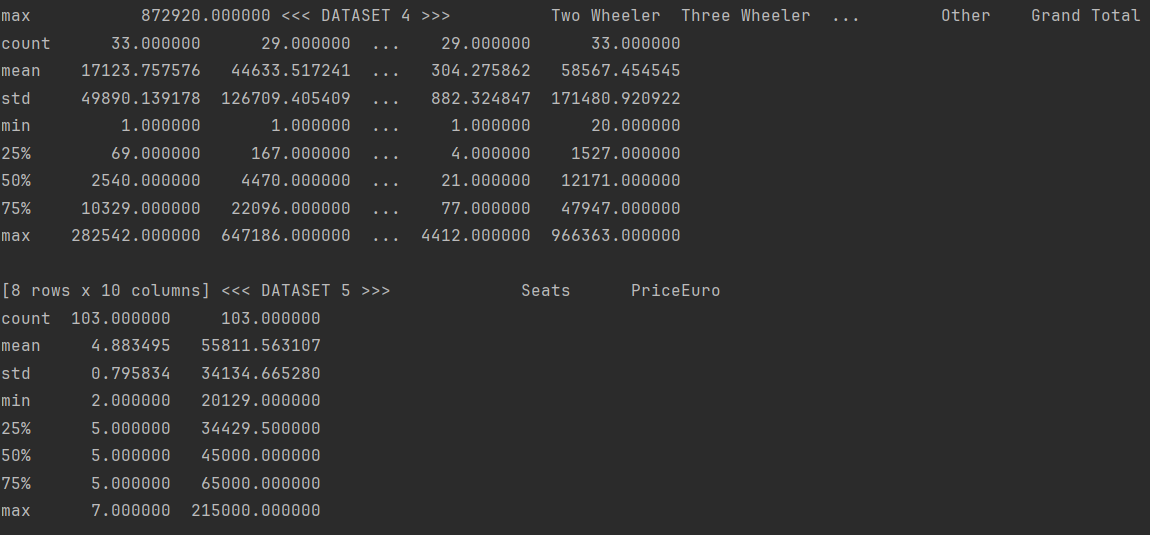
df3 = df3.describe()

df4 = df4.describe()

df5 = df5.describe()

print('<<< DATASET 1 >>>', df1, '<<< DATASET 2 >>>', df2, '<<< DATASET 3 >>>', df3, '<<< DATASET 4 >>>', df4, '<<< DATASET 5 >>>', df5)





**C.2 Analysis of 2-wheeler EVs**

# 2 wheelers data visualization from dataset 4

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

sns.barplot(data=df4, y=df4['State Name'].sort\_values(ascending=True), x='Two Wheeler', palette='viridis')

plt.ylabel('State', fontsize=14, family='serif')

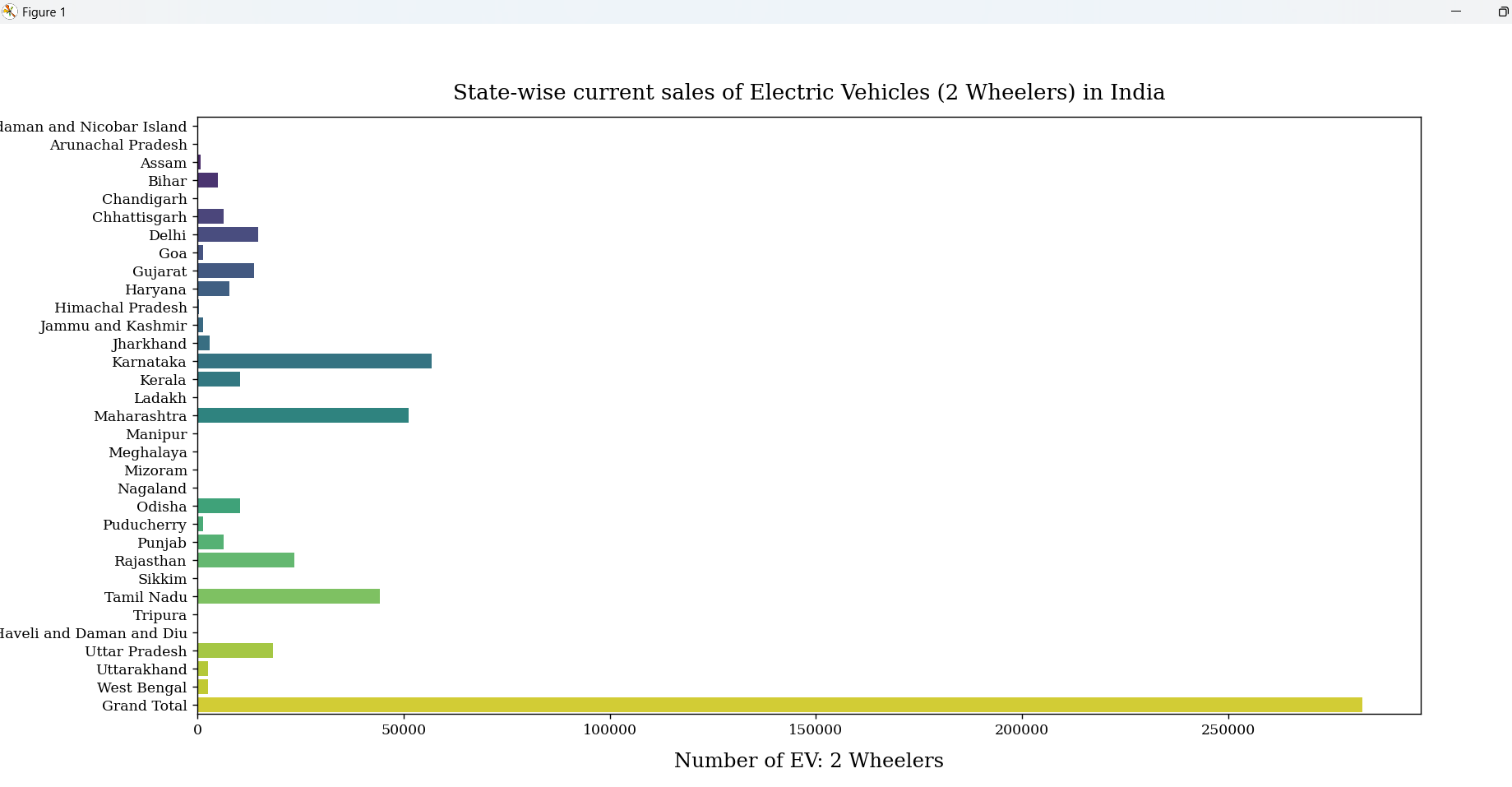
plt.xlabel('Number of EV: 2 Wheelers', family='serif', fontsize=14, labelpad=10)

plt.xticks(family='serif')

plt.yticks(family='serif')

plt.title(label='State-wise current sales of Electric Vehicles (2 Wheelers) in India', weight=200, family='serif', size=15, pad=12)

plt.show()



**C.3 Analysis of 3-Wheeler EVs**

# 3 wheelers data visualization from dataset 4

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

sns.barplot(data=df4, y=df4['State Name'].sort\_values(ascending=True), x='Three Wheeler', palette='viridis')

plt.ylabel('State', fontsize=14, family='serif')

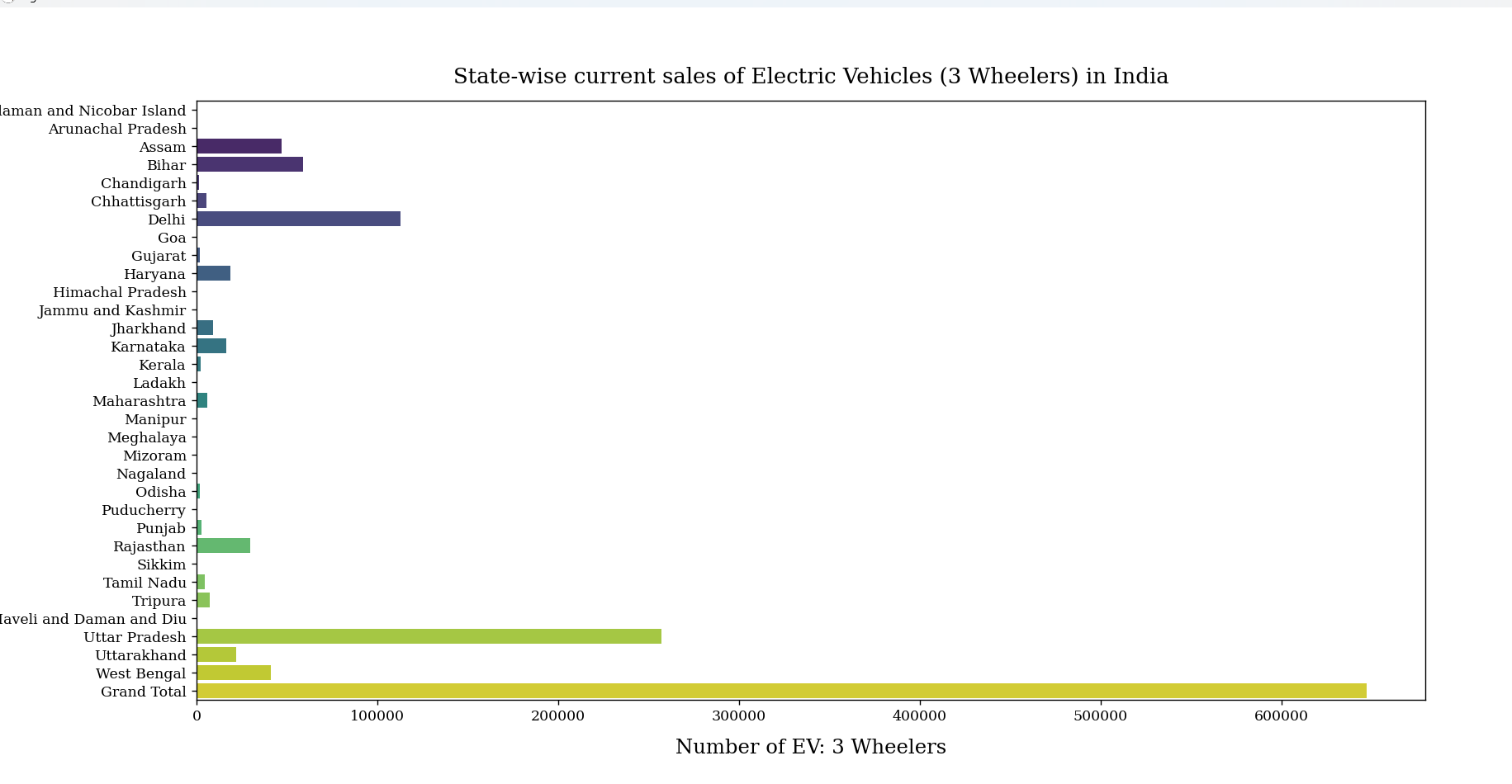
plt.xlabel('Number of EV: 3 Wheelers', family='serif', fontsize=14, labelpad=10)

plt.xticks(family='serif')

plt.yticks(family='serif')

plt.title(label='State-wise current sales of Electric Vehicles (3 Wheelers) in India', weight=200, family='serif', size=15, pad=12)

plt.show()



**C.4 Analysis of 4-wheeler EVs**

# 4 wheelers data visualization from dataset 4

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

sns.barplot(data=df4, y=df4['State Name'].sort\_values(ascending=True), x='Four Wheeler', palette='viridis')

plt.ylabel('State', fontsize=14, family='serif')

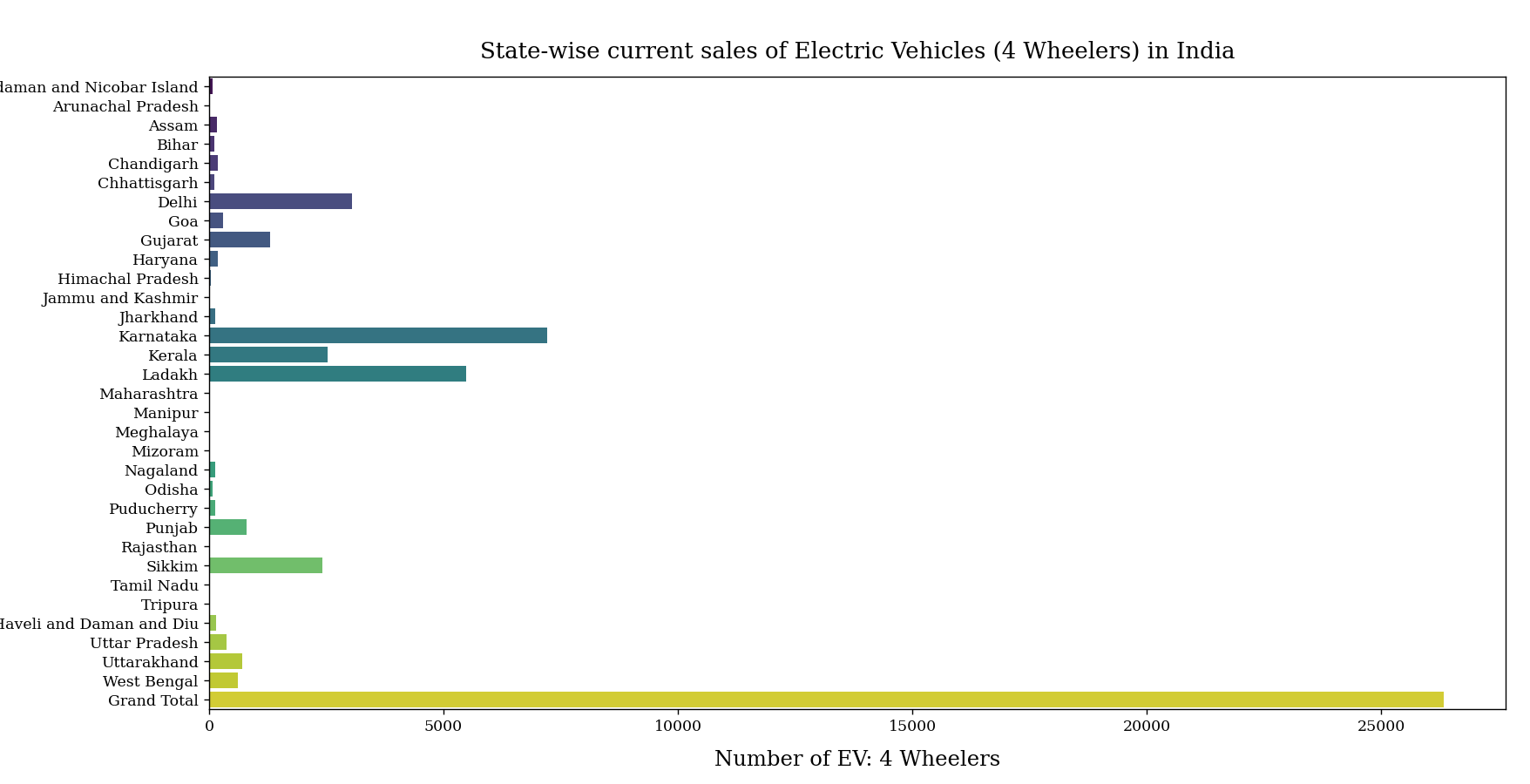
plt.xlabel('Number of EV: 4 Wheelers', family='serif', fontsize=14, labelpad=10)

plt.xticks(family='serif')

plt.yticks(family='serif')

plt.title(label='State-wise current sales of Electric Vehicles (4 Wheelers) in India', weight=200, family='serif', size=15, pad=12)

plt.show()



**C.5 Analysis of EV charging stations availability in India**

# charging stations availability visualization from dataset 2

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

sns.barplot(data=df2, y=df2['City/Highway'].sort\_values(ascending=True), x='Charging Stations', palette='viridis')

plt.ylabel('State', fontsize=14, family='serif')

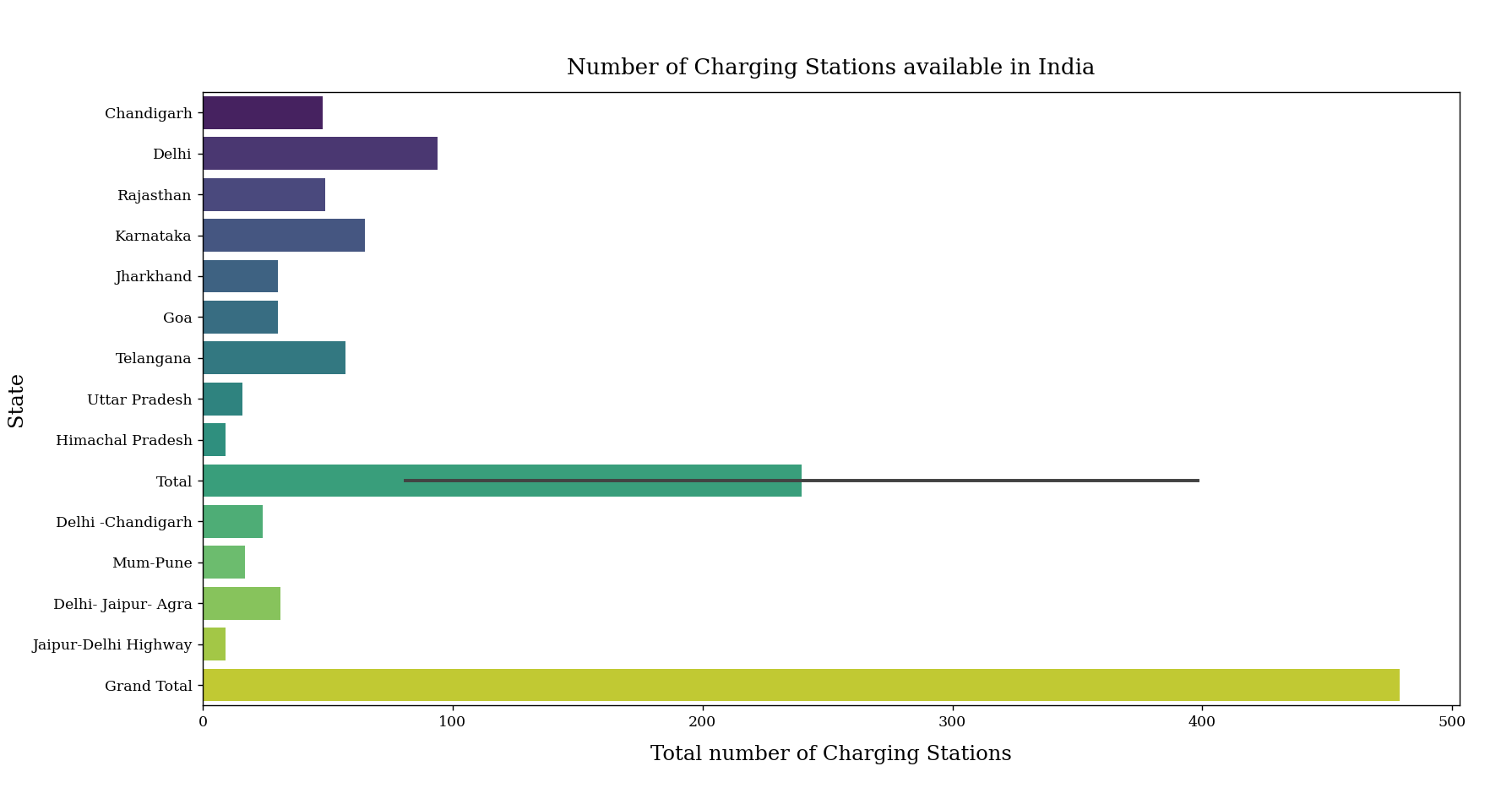
plt.xlabel('Total number of Charging Stations', family='serif', fontsize=14, labelpad=10)

plt.xticks(family='serif')

plt.yticks(family='serif')

plt.title(label='Number of Charging Stations available in India', weight=200, family='serif', size=15, pad=12)

plt.show()



**C.6 EVs based on brands**

# brand-wise count of EV models

sns.catplot(data=df5, x='Brand', kind='count', palette='viridis', height=6, aspect=2)

sns.despine(right=False, top=False)

plt.tick\_params(axis='x', rotation=40)

plt.xlabel('Brand',family='serif', size=12)

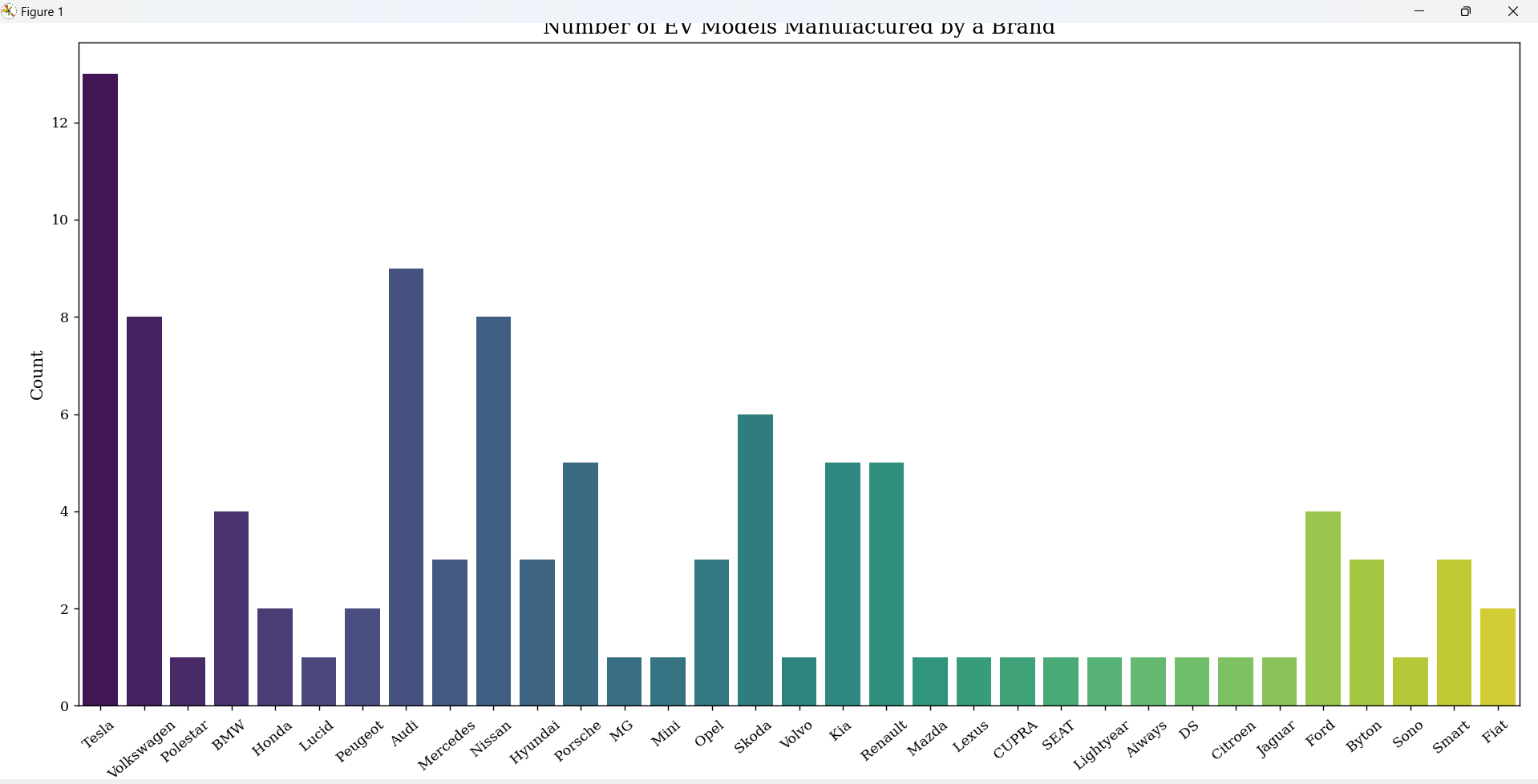
plt.ylabel('Count', family='serif', size=12)

plt.xticks(family='serif')

plt.yticks(family='serif')

plt.title('Number of EV Models Manufactured by a Brand', family='serif', size=15)

plt.show()



The top 5 electric vehicle manufacturers in Indian EV Market includes Tesla, Audi, Nissan, Volkswagen and Skoda.

**C.7 Analysis of different segments of EVs in India**

# analysis of different segments of EVs from dataset 5

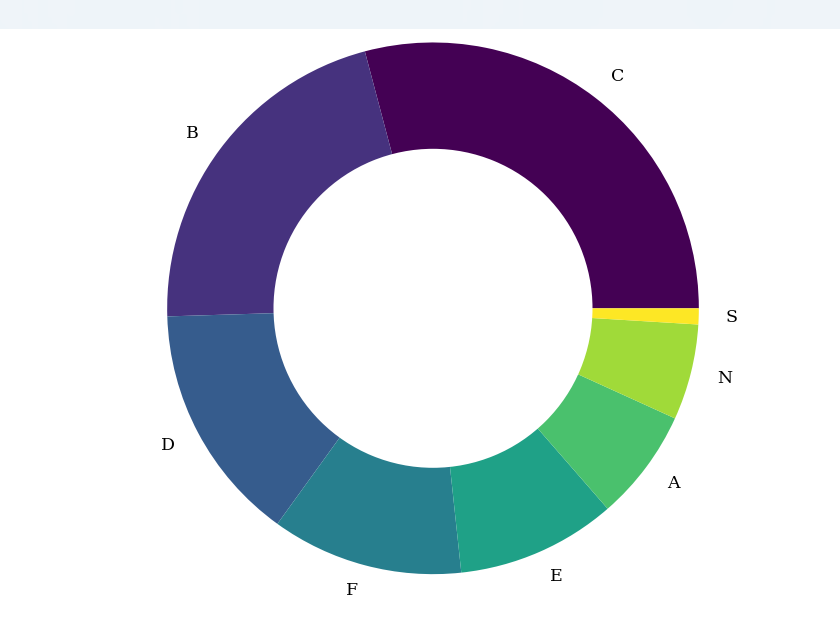
x = df5['Segment'].value\_counts().plot.pie(radius=2, cmap='viridis', startangle=0, textprops=dict(family='serif'), pctdistance=.5)

plt.pie(x=[1], radius=1.2, colors='white')

plt.title(label='Electric Vehicles of Different Segments in India', family='serif', size=15, pad=100)

plt.ylabel('')

plt.show()



B and C captures the broader segments among all while S and A captures the least among all of the others.

**C.8 Analysis of different body types present in different EV models in India**

# different body types EVs visualization from dataset 5

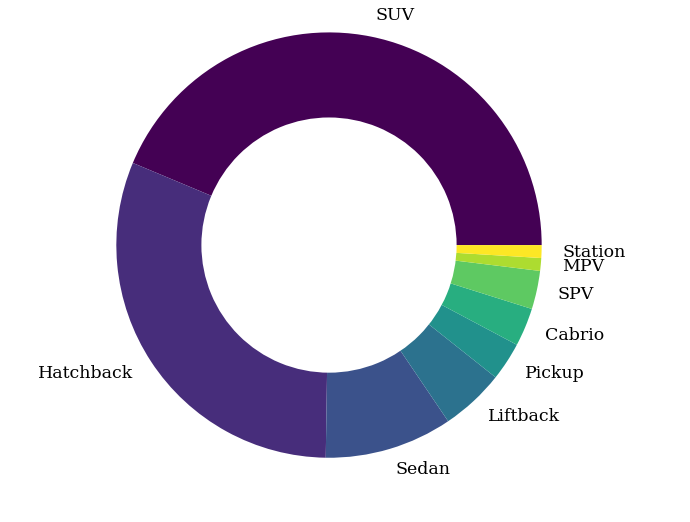
x = df5['BodyStyle'].value\_counts().plot.pie(radius=2, cmap='viridis', startangle=0, textprops=dict(family='serif'))

plt.pie(x=[1], radius=1.2, colors='white')

plt.title(label='Electric Vehicles of Different Body Types based on different EV Models in India', family='serif', size=15, pad=100)

plt.ylabel('')

plt.show()



C.9 Correlation between selling price and other variables of EV market in India

# Correlation between selling price and other variables and plotting of correlation heatmap from dataset - 1

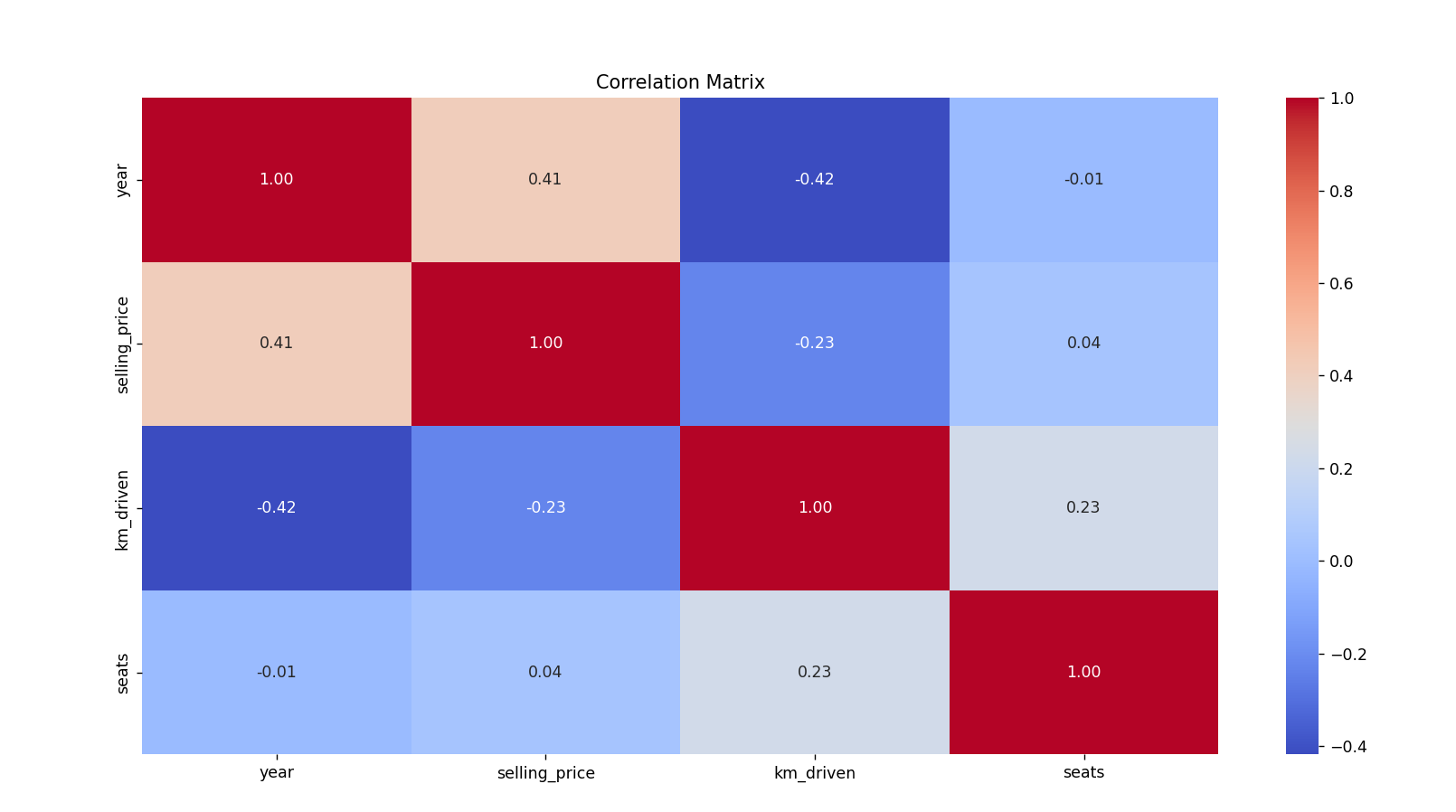
numeric\_columns = df1.select\_dtypes(include=['int', 'float']).columns

corr\_matrix = df1[numeric\_columns].corr()

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title('Correlation Matrix')

plt.show()



**C.10 Analysis of different number of seats by each brand**

# brand-wise analysis of the number of seats

sns.countplot(data=df5, x='Brand', hue='Seats', palette='viridis')

sns.despine(right=False, top=False)

plt.tick\_params(axis='x', rotation=40)

plt.xlabel('Brand',family='serif', size=12)

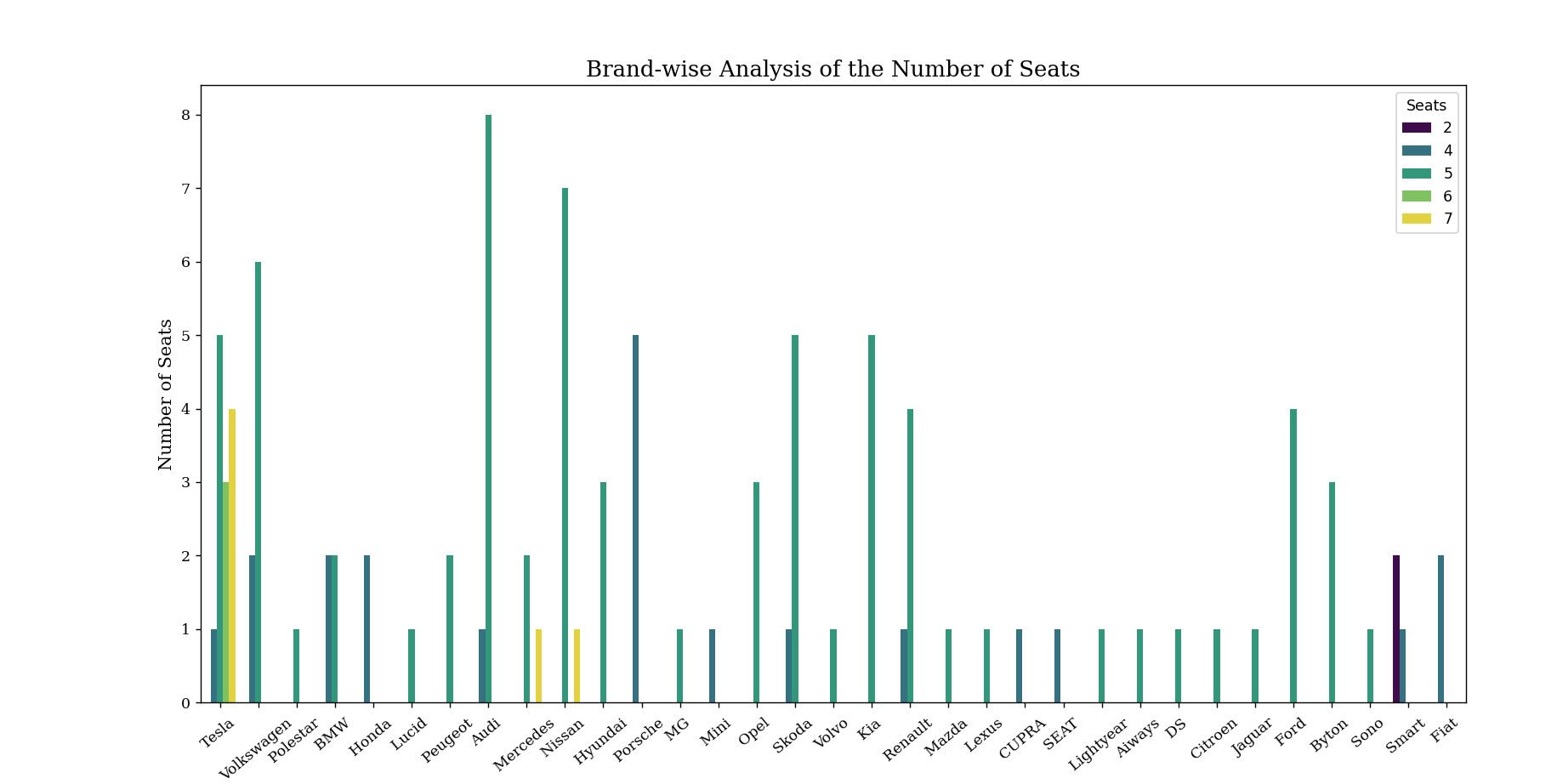
plt.ylabel('Number of Seats', family='serif', size=12)

plt.xticks(rotation=40, family='serif')

plt.yticks(family='serif')

plt.title('Brand-wise Analysis of the Number of Seats', family='serif', size=15)

plt.show()



**C.11 Analysis of EVs based on speed**

# speed visualization from dataset 5

plt.figure(figsize=(6, 8))

sns.barplot(data=df5, x='TopSpeed', hue='Brand', palette='viridis')

plt.xticks(family='serif')

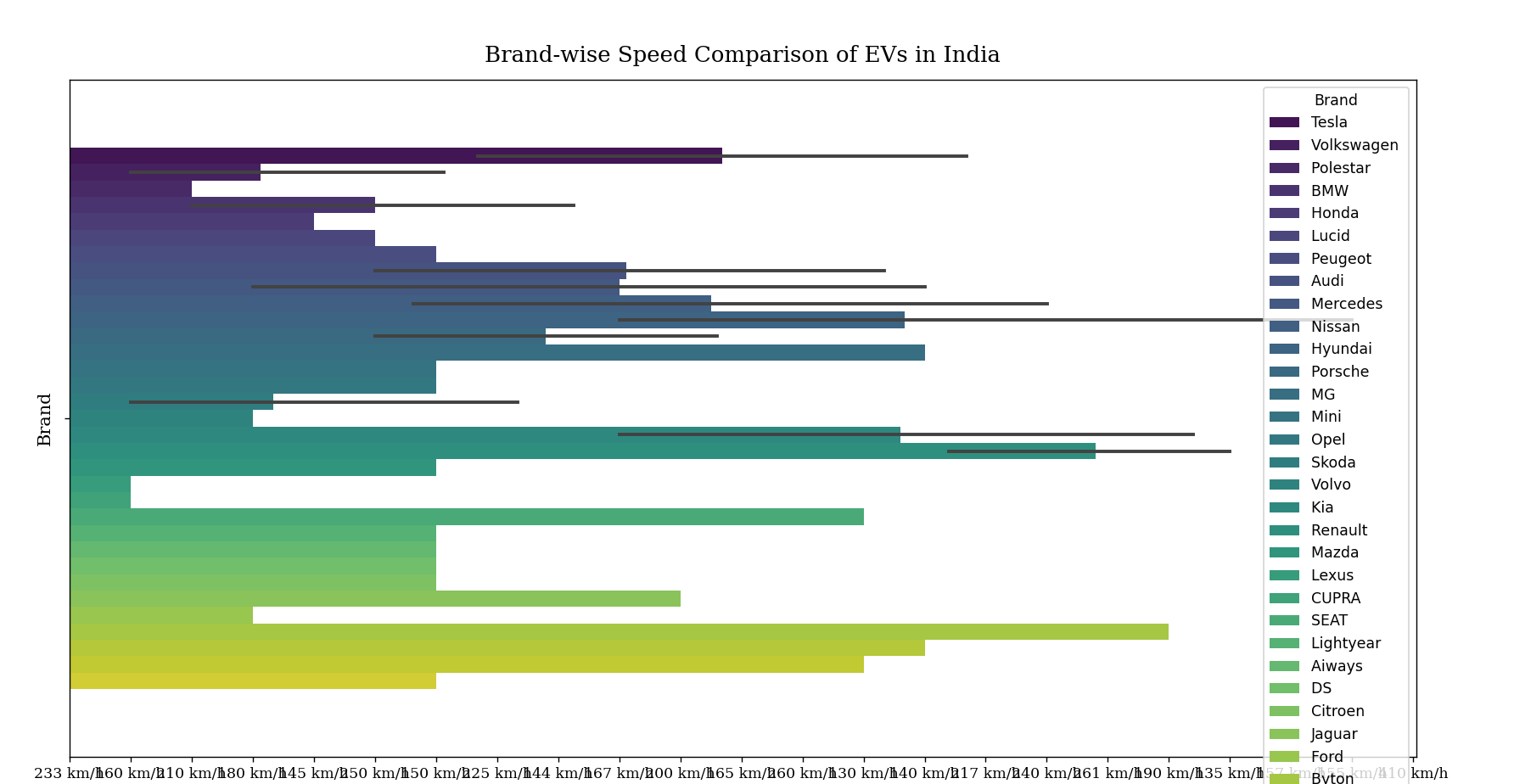
plt.yticks(family='serif')

plt.xlabel('Max Speed', family='serif', size=12)

plt.ylabel('Brand', family='serif', size=12)

plt.title(label='Brand-wise Speed Comparison of EVs in India', family='serif', size=15, pad=12)

plt.show()



**C.12 Analysis of EVs based on Acceleration**

# acceleration visualization from dataset 5

plt.figure(figsize=(6, 8))

sns.barplot(data=df5, x='Accel', hue='Brand', ci=None, palette='viridis')

plt.xticks(family='serif')

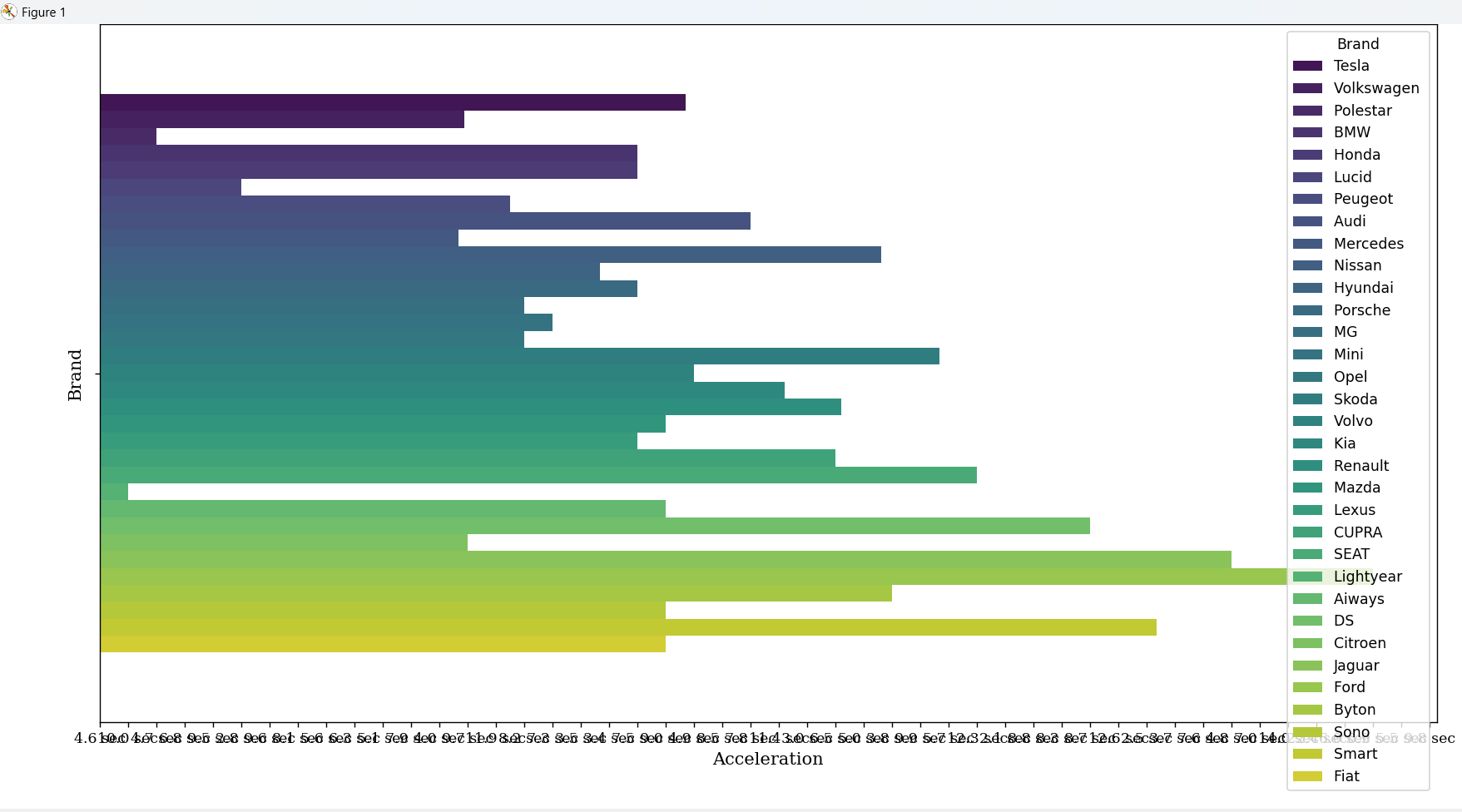
plt.yticks(family='serif')

plt.xlabel('Acceleration', family='serif', size=12)

plt.ylabel('Brand', family='serif', size=12)

plt.title(label='Acceleration of EVs in India', family='serif', size=15, pad=12)

plt.show()



**C.13 Analysis of EV brands/models based on range parameters**

# brand-wise analysis of the range parameter

plt.figure(figsize=(8, 6))

sns.barplot(data=df5, x='Brand', hue ='Range', palette='viridis', ci=None)

sns.despine(right=False, top=False)

plt.xticks(rotation=40, ha='right', family='serif', size=10)

plt.yticks(family='serif', size=10)

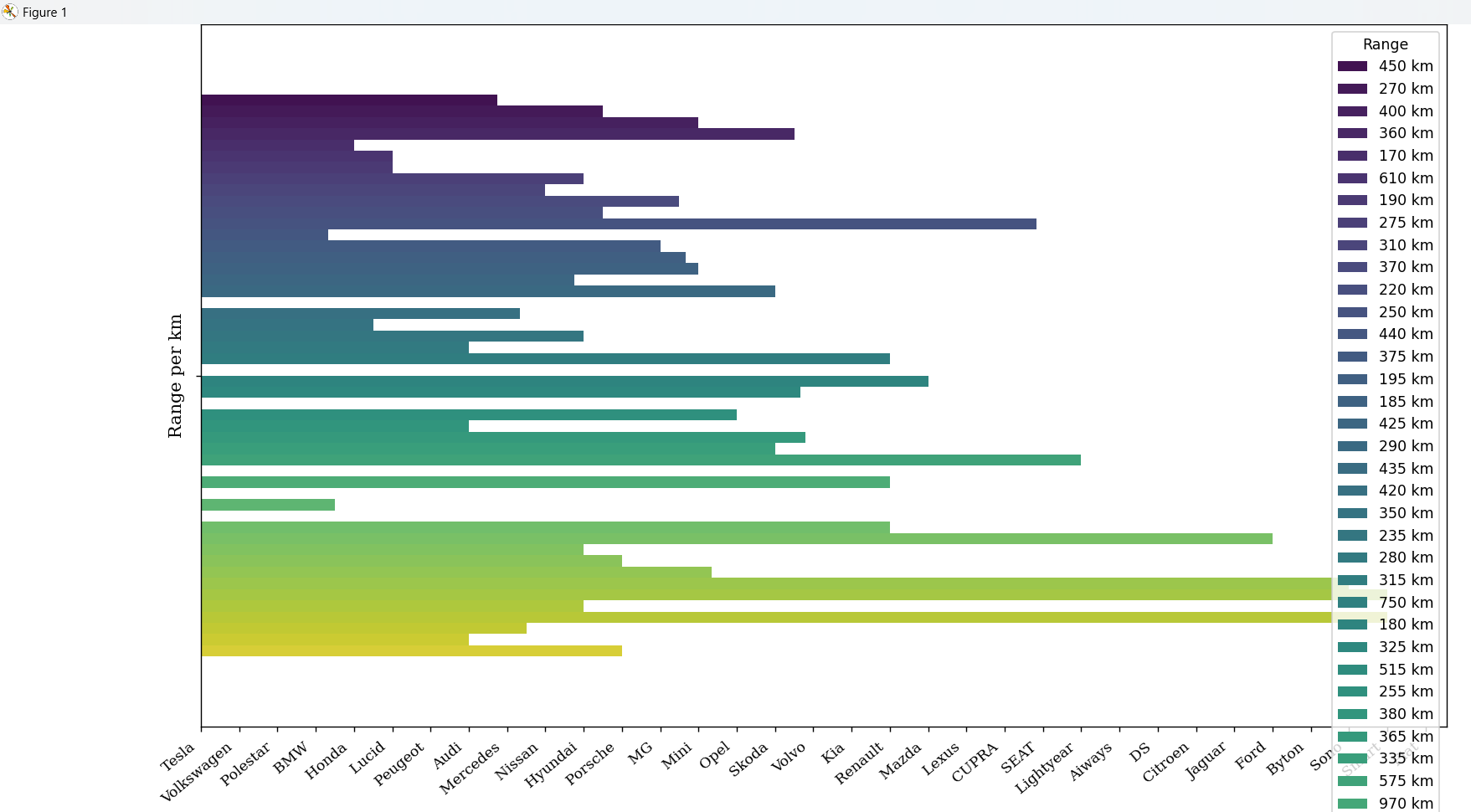
plt.xlabel('Brand',family='serif', size=12)

plt.ylabel('Range per km', family='serif', size=12)

plt.title('Brand-wise Analysis of the Range Parameter', family='serif', size=15)

plt.tight\_layout()

plt.show()



1. **Segmentation Approach**

**Decision Trees Algorithm**

Over here, each decision node represents a feature and the branches represent the decision paths based on the values of those features. The decision tree algorithm evaluates the feature importance based on how well each feature splits the data. The decision tree then selects the feature with the highest importance.

The decision tree algorithm works the following way:

1. At first, it begins with the root node i.e. (initial decision)
2. It then chooses a feature to split the data based on certain criteria.
3. Split the data into subsets based on the chosen feature
4. Steps 2 and 3 are being repeated for each subset (child node) recursively, until a stopping criterion is met.
5. It then determines the feature importance based on the decision tree structure
6. Select the features with the highest importance for building the model

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean\_absolute\_error

# Handling missing values

df5.replace('-', np.nan, inplace=True) # Replace '-' with NaN for missing values

df5.dropna(inplace=True)

#Encoding the categorical features

df5['PowerTrain'].replace(to\_replace=['RWD', 'FWD', 'AWD'], value=[0, 1, 2], inplace=True)

# RapidCharge feature

df5['RapidCharge'].replace(to\_replace=['No', 'Yes'], value=[0, 1], inplace=True)

# Selecting features for building the model

X = df5[

['Accel\_Sec', 'TopSpeed\_KmH', 'Efficiency\_WhKm', 'FastCharge\_KmH', 'Range\_Km', 'RapidCharge', 'Seats', 'PowerTrain',

'PriceEuro']]

y = df5['PriceEuro'] # Target variable

# Feature scaling

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

depths = []

mae\_scores = []

# Iterate over different depths of decision tree

for depth in range(1, 11):

# Initialize and train the decision tree regressor

clf = DecisionTreeRegressor(max\_depth=depth, random\_state=42)

clf.fit(X\_train, y\_train)

# Make predictions on the testing set

y\_pred = clf.predict(X\_test)

# Calculate MAE and store in lists

depths.append(depth)

mae = mean\_absolute\_error(y\_test, y\_pred)

mae\_scores.append(mae)

# Plot the results

plt.figure(figsize=(8, 6))

plt.plot(depths, mae\_scores, marker='o', linestyle='-')

plt.title('Decision Tree MAE vs. Depth', size=15, family='serif')

plt.xlabel('Depth of Decision Tree', family='serif')

plt.ylabel('Mean Absolute Error', family='serif')

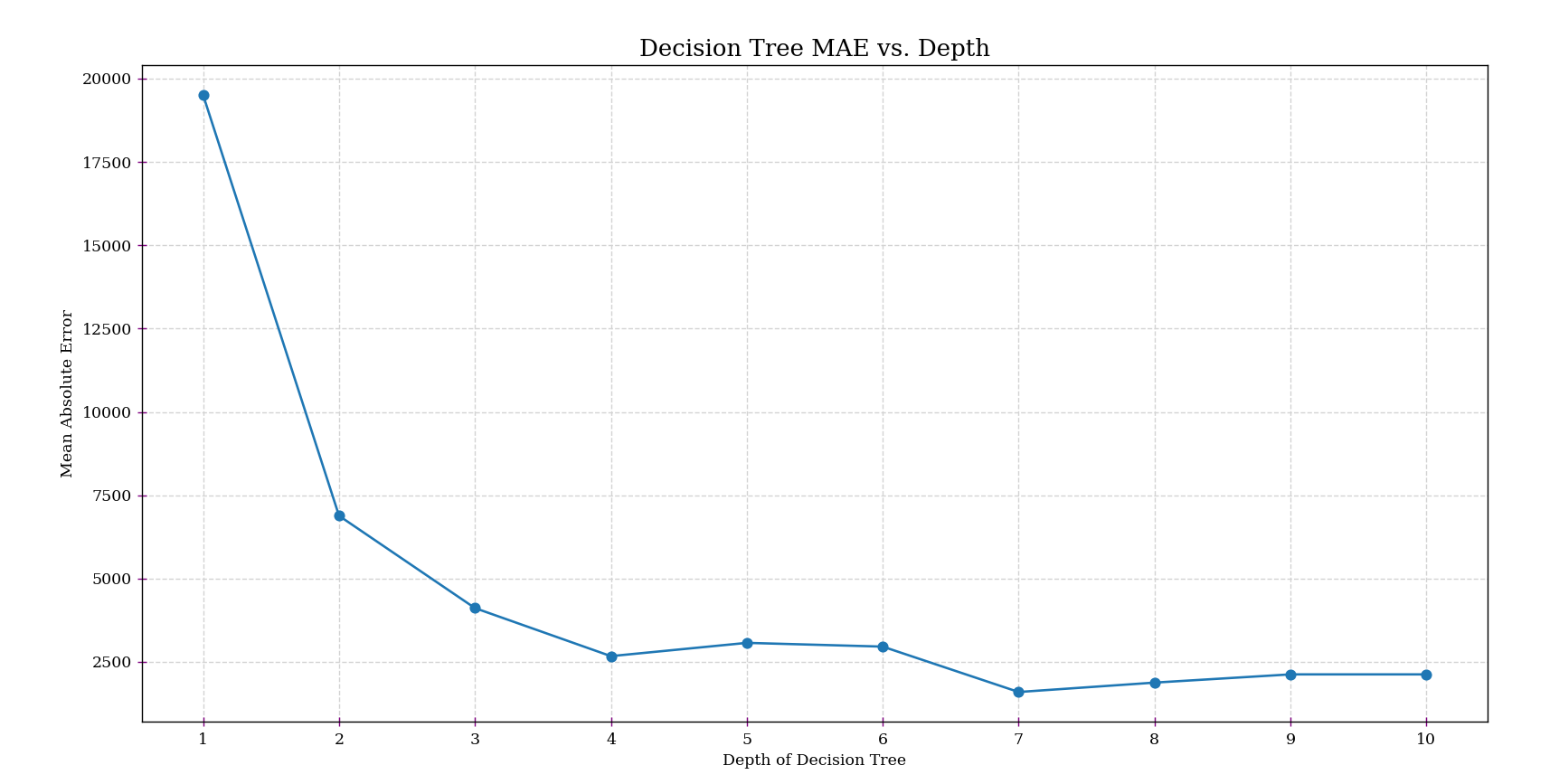
plt.xticks(range(1, 11), family='serif')

plt.yticks(family='serif')

plt.grid()

plt.tick\_params(axis='both', direction='inout', length=6, color='purple', grid\_color='lightgray', grid\_linestyle='--')

plt.show()



By using decision tree algorithm, it helps in identification of missing values, handles missing values, converts the categorical variables etc. In this context, regression evaluation metric**, mean absolute error** has been used. The regression metric has been used instead of a categorical evaluation metric since my target variable ‘y’ is continuous, whereby ‘y’ represents the price of electric vehicles. Prices can vary continuously, with no limit of distinct categories between the values i.e. there is no such limited range that can be considered. Therefore, ‘y’ may contain a mix of continuous and categorical values.

The above visualization graph shows the decision tree model at different depths:

**a) X-axis (Depth of decision tree):** The depth controls how many splits the decision tree can make during training.

**b) Y-axis (Mean Absolute Error):** This represents the average absolute difference between the actual prices of electric vehicles and the prices predicted by the decision tree model. MAE measures the average magnitude of errors in the predictions. Lower values of MAE indicate better performance, as they mean the model's predictions of prices are closer to the actual prices as given in the dataset-5.

Here, we are looking for the point where Mean Absolute Error is the lowest. It is the lowest at x=7.000 and y=1630. The lowest value indicates the optimal depth of the decision tree for the dataset-5.