CHAPTER 4

INTRODUCTION

4.1 Overview

This section discusses the analysis and prediction of the market trends for various types of electric vehicles. This chapter will start with the identification of datasets, analyze datasets, establish historical trend models, and use machine learning techniques to build the results of the model. The mechanical learning techniques used include Long Short-Term Memory (LSTM), Deep Learning Models, and Hybrid Models. The results of the predictions based on these machine learning techniques on the trends in the EV market are presented in the following sections.

4.2 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a method for exploring and understanding data that can help us spot data characteristics, trends, and outliers. When predicting the development trend of the electric vehicle market, the detailed analysis and processing of the data using the EDA method is helpful to make a more accurate prediction of the development trend of the electric vehicle market, and the EDA can be carried out through the following steps:

```
df
```

	region	category	parameter	mode	powertrain	year	unit	value	percentage
0	Austria	Historical	EV stock	Cars	BEV	2010	Vehicles	350	35000,00%
1	Austria	Historical	EV stock share	Cars	EV	2010	percent	789.999.961.853	78999996185300,00%
2	Belgium	Historical	EV stock	Buses	BEV	2010	Vehicles	3	300,00%
3	Belgium	Historical	EV sales	Vans	BEV	2010	Vehicles	7	700,00%
4	Belgium	Historical	EV stock	Vans	BEV	2010	Vehicles	62	6200,00%
...	...	...	...	...	...	...	...	...	...
12649	World	Projection-STEPS	EV sales share	Cars	EV	2035	percent	55	5500,00%
12650	World	Projection-STEPS	EV stock share	Cars	EV	2035	percent	31	3100,00%
12651	World	Projection-APS	EV charging points	EV	Publicly available fast	2035	charging points	9400000	940000000,00%
12652	World	Projection-APS	EV charging points	EV	Publicly available slow	2035	charging points	15000000	1500000000,00%
12653	World	Projection-STEPS	EV stock share	Trucks	EV	2035	percent	9	900,00%

12654 rows × 9 columns

Figure 4.1 Dataset

As shown in the figure above, the source dataset used in this study includes a total of 12,654 rows and 9 columns.

```
# One-hot encode categorical variables
X = pd.get_dummies(df.drop(columns=['value']), columns=['region', 'category', 'parameter', 'mode', 'powertrain', 'unit'])

# Separate the target variable
y = df['value']

print(X.head())   # Check the one-hot encoded features
```

```
   year    percentage  region_Australia  region_Austria  region_Belgium  \
0  2010  3.500000e+06             False            True           False
1  2010  7.900000e+15             False            True           False
2  2010  3.000000e+04             False           False            True
3  2010  7.000000e+04             False           False            True
4  2010  6.200000e+05             False           False            True

   region_Brazil  region_Bulgaria  region_Canada  region_Chile  region_China  \
0          False            False          False         False         False
1          False            False          False         False         False
2          False            False          False         False         False
3          False            False          False         False         False
4          False            False          False         False         False

   ...  powertrain_FCEV  powertrain_PHEV  powertrain_Publicly available fast  \
0  ...            False            False                               False
1  ...            False            False                               False
2  ...            False            False                               False
3  ...            False            False                               False
4  ...            False            False                               False

   powertrain_Publicly available slow  unit_GWh  unit_Milion barrels per day  \
0                               False     False                        False
1                               False     False                        False
2                               False     False                        False
3                               False     False                        False
4                               False     False                        False

   unit_Oil displacement, million lge  unit_Vehicles  unit_charging points  \
0                               False           True                 False
1                               False          False                 False
2                               False           True                 False
3                               False           True                 False
4                               False           True                 False
```

Figure 4.2 Encode Categorical Variables

Encode Categorical Variables are coded for subsequent data analysis cleanup.

```
import numpy as np

# Check for NaN values
print("NaN values in X_train:", np.isnan(X_train).sum())
print("NaN values in X_test:", np.isnan(X_test).sum())

# Check for infinite values
print("Infinite values in X_train:", np.isinf(X_train).sum())
print("Infinite values in X_test:", np.isinf(X_test).sum())
```

```
NaN values in X_train: year                                0
region_Australia                0
region_Austria                  0
region_Belgium                  0
region_Brazil                   0
                               ..
unit_Milion barrels per day     0
unit_Oil displacement, million lge    0
unit_Vehicles                   0
unit_charging points            0
unit_percent                    0
Length: 83, dtype: int64
NaN values in X_test: year                                 0
region_Australia                0
region_Austria                  0
region_Belgium                  0
region_Brazil                   0
                               ..
unit_Milion barrels per day     0
unit_Oil displacement, million lge    0
unit_Vehicles                   0
unit_charging points            0
unit_percent                    0
Length: 83, dtype: int64
Infinite values in X_train: year                                0
region_Australia                0
region_Austria                  0
region_Belgium                  0
region_Brazil                   0
                               ..
unit_Milion barrels per day     0
unit_Oil displacement, million lge    0
unit_Vehicles                   0
```

Figure 4.3 Check for Missing or Infinite Values

Ensure that there are no missing (NaN) or infinite values in your training and test datasets before training the model.

```
[23] from sklearn.model_selection import train_test_split

# 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)
```

Figure 4.4 Split the Dataset

```
[25] from sklearn.preprocessing import StandardScaler

     # Standardize the features
     scaler = StandardScaler()
     X_train = scaler.fit_transform(X_train)
     X_test = scaler.transform(X_test)
```

Figure 4.5 Standardize the Features

```
[27] # Reshape data to fit the 1D CNN input
     X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
     X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

     df
```

	region	category	parameter	mode	powertrain	year	unit	value
0	Australia	Historical	EV stock share	Cars	EV	2011	percent	3.900000e-04
1	Australia	Historical	EV sales share	Cars	EV	2011	percent	6.500000e-03
2	Australia	Historical	EV sales	Cars	BEV	2011	Vehicles	4.900000e+01
3	Australia	Historical	EV stock	Cars	BEV	2011	Vehicles	4.900000e+01
4	Australia	Historical	EV stock	Cars	BEV	2012	Vehicles	2.200000e+02
...	...	...	...	...	...	...	...	...
12649	World	Projection-STEPS	EV sales share	Cars	EV	2035	percent	5.500000e+01
12650	World	Projection-STEPS	EV stock share	Cars	EV	2035	percent	3.100000e+01
12651	World	Projection-APS	EV charging points	EV	Publicly available fast	2035	charging points	9.400000e+06
12652	World	Projection-APS	EV charging points	EV	Publicly available slow	2035	charging points	1.500000e+07
12653	World	Projection-STEPS	EV stock share	Trucks	EV	2035	percent	9.000000e+00

12654 rows × 8 columns

Figure 4.6 Reshape Data for 1D CNN

Figure 4.4, Figure 4.5, and Figure 4.5 show the steps including Split the dataset into training and testing sets and standardize the features to ensure that they are on a similar scale. Then, reshape the data to be compatible with a 1D CNN.

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, Dense, Flatten, Dropout

# Build the model
model = Sequential()

# Add a 1D convolutional layer
model.add(Conv1D(filters=64, kernel_size=2, activation='relu', input_shape=(X_train.shape[1], 1)))

# Add a Flatten layer
model.add(Flatten())

# Add a Dense layer
model.add(Dense(128, activation='relu'))

# Add a Dropout layer for regularization
model.add(Dropout(0.2))

# Add the output layer
model.add(Dense(1, activation='linear'))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Display the model's architecture
model.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential_2"
```

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 82, 64)	192
flatten (Flatten)	(None, 5248)	0
dense (Dense)	(None, 128)	671,872
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

Total params: 672,193 (2.56 MB)
Trainable params: 672,193 (2.56 MB)
Non-trainable params: 0 (0.00 B)

Figure 4.7 Build the 1D CNN Model

This step builds a simple 1D CNN model for regression.

```
[ ] # Train the model
history = model.fit(X_train, y_train, epochs=50, validation_data=(X_test, y_test), batch_size=32)

317/317 ━━━━━━━━━━━━━━━━━━━━ 9s 12ms/step - loss: 66445782286336.0000 - val_loss: 837007924264
Epoch 23/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 7s 19ms/step - loss: 55054304280576.0000 - val_loss: 836685801717
Epoch 24/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - loss: 27219581730816.0000 - val_loss: 836326349864
Epoch 25/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - loss: 80761742950400.0000 - val_loss: 835895343185
Epoch 26/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - loss: 34965416837120.0000 - val_loss: 835476416102
Epoch 27/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 6s 19ms/step - loss: 45324504662016.0000 - val_loss: 835085171425
Epoch 28/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 9s 27ms/step - loss: 37333755756544.0000 - val_loss: 834671948595
Epoch 29/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 7s 15ms/step - loss: 55015263698944.0000 - val_loss: 834229030092
Epoch 30/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - loss: 59982254964736.0000 - val_loss: 833770844323
Epoch 31/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 4s 14ms/step - loss: 25464619925504.0000 - val_loss: 833280362414
Epoch 32/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 47399196164096.0000 - val_loss: 832762198097
Epoch 33/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - loss: 27119182675968.0000 - val_loss: 832242523832
Epoch 34/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 6s 14ms/step - loss: 55144095940608.0000 - val_loss: 831709008363
Epoch 35/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 6s 17ms/step - loss: 28061691019264.0000 - val_loss: 831179435540
Epoch 36/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - loss: 41846394847232.0000 - val_loss: 830633672704
Epoch 37/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - loss: 20842754867200.0000 - val_loss: 830065512284
Epoch 38/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 6s 20ms/step - loss: 22580278853632.0000 - val_loss: 829533926195
Epoch 39/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - loss: 30645900279808.0000 - val_loss: 828951253483
Epoch 40/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - loss: 27313483808768.0000 - val_loss: 828381499228
Epoch 41/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 7s 19ms/step - loss: 18212183867392.0000 - val_loss: 827776932249
Epoch 42/50
317/317 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - loss: 85511347634176.0000 - val_loss: 827213050019
```

Figure 4.8 Train the Model

Train the model using the training data.

```
# Evaluate the model
test_loss = model.evaluate(X_test, y_test)
print(f'Test Loss: {test_loss}')

80/80 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: nan
Test Loss: nan
```

Figure 4.9 Evaluate the Model

```
[ ] import matplotlib.pyplot as plt

    # Plot the training and validation loss
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
```

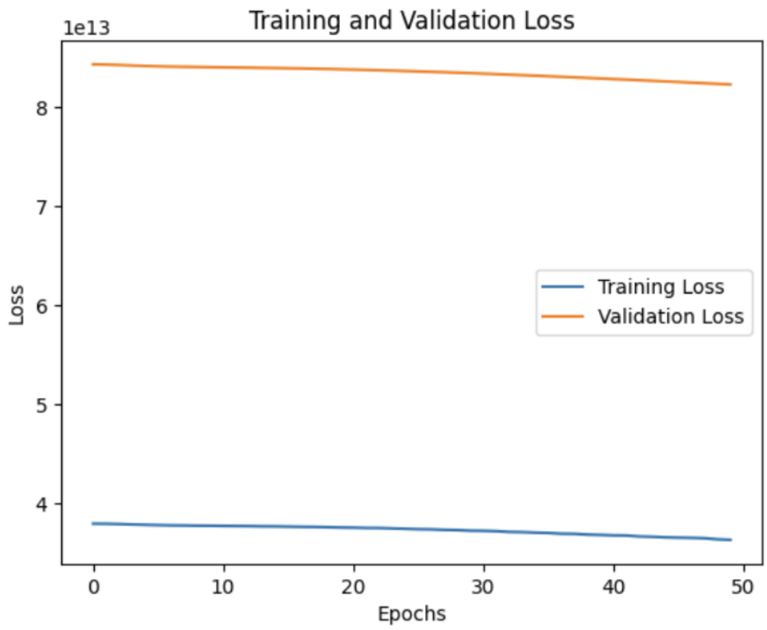

Figure 4.10 Visualize the Training Process

Visualize the training and validation loss to understand how well the model is training.

4.3 Explore the trend of total annual sales of electric vehicles

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

a, b, c, d = plt.cm.Reds, plt.cm.Greens, plt.cm.Blues, plt.cm.Purples
mode = ['Buses', 'Cars', 'Trucks', 'Vans']
group_mode = data_sale.groupby(['mode'])['value'].sum()

# Convert 'value' column to numeric before grouping
data_sale['value'] = pd.to_numeric(data_sale['value'], errors='coerce')
group_mode = data_sale.groupby(['mode'])['value'].sum()

# Convert values to numeric, handling potential errors
mode_num = []
for i in mode:
    try:
        mode_num.append(int(group_mode[i]))
    except ValueError:
        print(f"Warning: Could not convert value for mode '{i}' to int. Using 0 instead.")
        mode_num.append(0)  # Or handle the error differently

plt.figure(figsize=(6, 6), dpi=300)
explode = [0.02, 0.02, 0.02, 0.02]
pie1, _, _ = plt.pie(mode_num, labels=mode, autopct='%1.2f%%', pctdistance=1.1,
                     labeldistance=0.8, radius=1.2, colors=[a(0.6), b(0.6), c(0.6), d(0.6)]
                     explode=explode, textprops={'fontsize': 10})
plt.setp(pie1, width=0.5, edgecolor='k')
plt.title('Proportion of various mode vehicles sales', fontsize=20)
plt.show()
```

Figure 4.11 Analyse the proportional composition of sales of vehicles

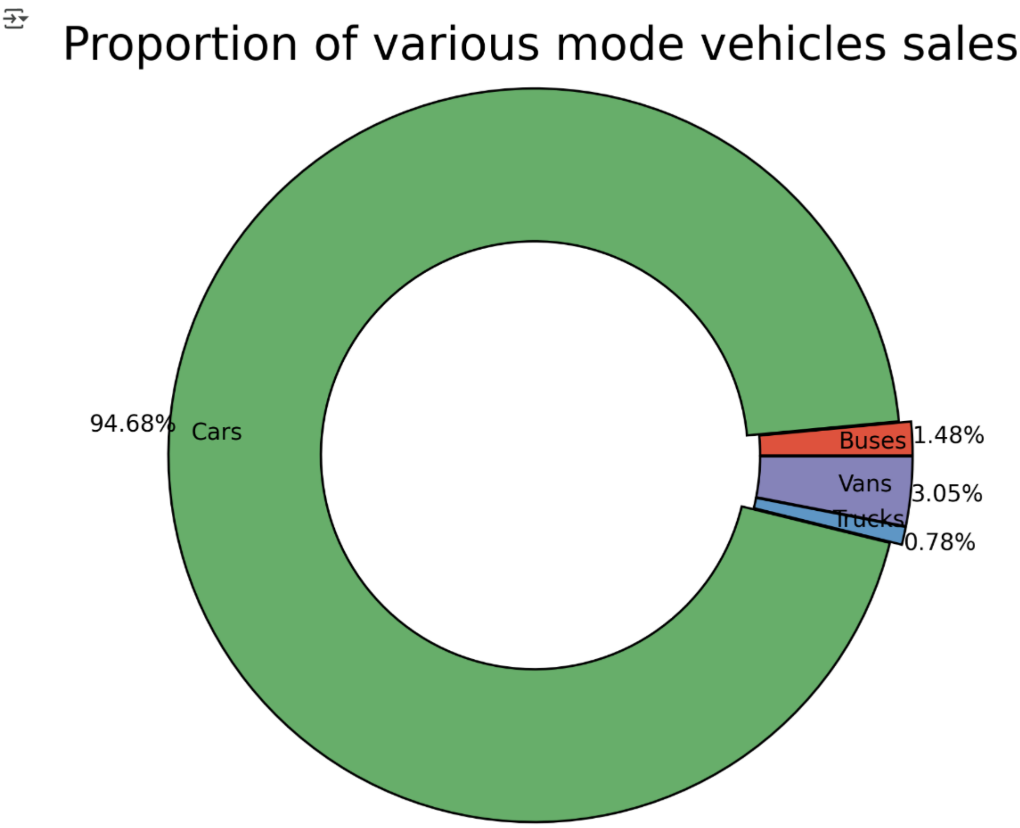

Figure 4.12 Proportion of various mode vehicles sales

As can be seen from the above chart, among the electric vehicles sold, cars account for the overwhelming majority, followed by minivans, buses, and trucks. Next, we will analyze the proportion of vehicle sales by different powertrains.

```
group_powertrain=data_sale.groupby('powertrain')['value'].sum()
plt.figure(figsize=(6,6),dpi=300)
explode=[0.02,0.02,0.02,]
pie,_,_=plt.pie(group_powertrain,labels=group_powertrain.index,autopct='%1.2f%%',pctdistance=1.1,labeldistance=0.8,ra
                            explode=explode)
plt.setp(pie,width=0.5,edgecolor='k')
plt.title('Proportion  of  various  powertrain  vehicles  sales',fontsize=20)
plt.show()
```

Figure 4.13 Analyze the proportional composition of sales of vehicles

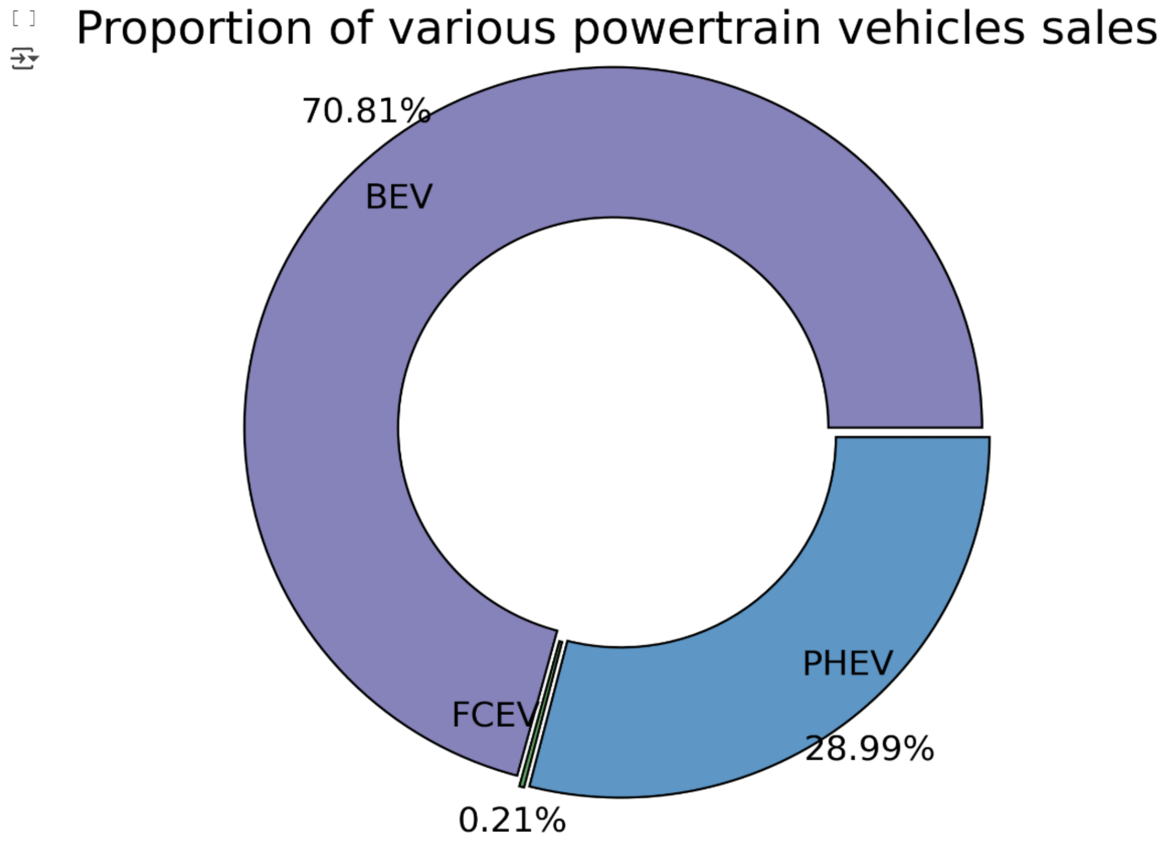

Figure 4.14 Proportion of various powertrain vehicles sales

As shown in the above chart, the sales proportion of Battery Electric Vehicles (BEV) accounts for more than one-third, Plug-in Hybrid Electric Vehicles (PHEV) are slightly lower than one-third, and Fuel Cell Electric Vehicles (FCEV) are negligible. Next, we will analyse the annual sales and proportion of Battery Electric Vehicles (BEV) and Plug-in Hybrid Electric Vehicles (PHEV).

```python
group_year_power=data_sale.groupby(['year','powertrain'])['value'].sum()
group_year_power=group_year_power.unstack(level=-1)
group_year_power['sum']=group_year_power.sum(axis=1)
xaxis=group_year_power.index
bev=group_year_power['BEV'].astype(int)
phev=-group_year_power['PHEV'].astype(int)
plt.figure(figsize=(9,6),dpi=300,constrained_layout=True)
plt.bar(xaxis,bev,color='yellowgreen',label='Battery Electric Vehicle')
plt.bar(xaxis,phev,color='palevioletred',label='Plug-in hybrid electric vehicle')
plt.legend(loc='best')
plt.yticks([])
for i ,j in zip(xaxis,bev):
        plt.annotate(j,(i,j),fontsize=10,ha='center')
for i ,j in zip(xaxis,phev):
        plt.annotate(-j,(i,j-500000),fontsize=10,ha='center')
plt.xticks(xaxis)
plt.xlabel('Year',fontsize=20)
plt.ylabel('Sales',fontsize=20)
plt.title('Annual various powertrain vehicles sales',fontsize=20)
plt.show()
```

Figure 4.15 Analyze the trends of electric vehicles.

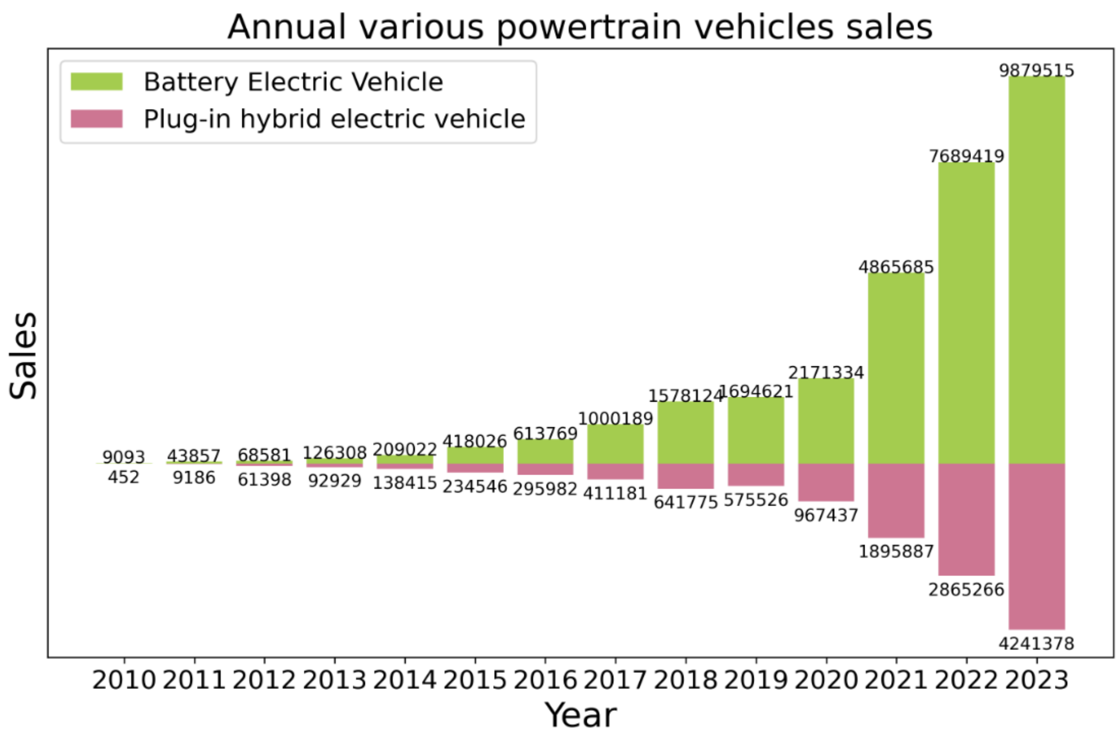

Figure 4.16 Chart of Annual various powertrain vehicles sales

This graph analyses the annual sales of various powertrain vehicles between 2010 and 2023, especially for battery electric vehicles and plug-in hybrid electric vehicles, during which sales of battery electric vehicles increased significantly, especially after 2018, when the growth rate accelerated. In 2023, sales of battery electric vehicles reached 9879515 units. In contrast, sales of plug-in hybrid electric vehicles are also increasing, but at a relatively slower pace, with 4241378 units sold in 2023.

```
[ ] fig,ax=plt.subplots(figsize=(9,6),dpi=300,constrained_layout=True)
    ax.set_xlim(0,100)
    bev_ratio=group_year_power['BEV']/group_year_power['sum']*100
    fcev_ratio=group_year_power['FCEV']/group_year_power['sum']*100
    phev_ratio=group_year_power['PHEV']/group_year_power['sum']*100
    ax.barh(xaxis,bev_ratio,color='yellowgreen',label='BEV')
    ax.barh(xaxis,fcev_ratio,left=bev_ratio,color='cornflowerblue',label='FCEV')
    ax.barh(xaxis,phev_ratio,left=bev_ratio+fcev_ratio,color='palevioletred',label='PHEV')
    for i,j in zip(bev_ratio,xaxis):
            ax.annotate(f'{i:.2f}%',(i/2,j),va='center')
    for i,j in zip(phev_ratio,xaxis):
            ax.annotate(f'{i:.2f}%',(100-i+i/2,j),va='center')
    ax.set_yticks(xaxis)
    ax.set_ylim(2009,2025)
    ax.set_xlabel('Sales percent')
    ax.set_ylabel('Year')
    ax.set_title('Annual various powertrain vehicles sales proportion',fontsize=20)
    ax.legend(loc='upper left',ncols=3)
    plt.show()
```

Figure 4.17 Analyze Annual various powertrain vehicles sales proportion

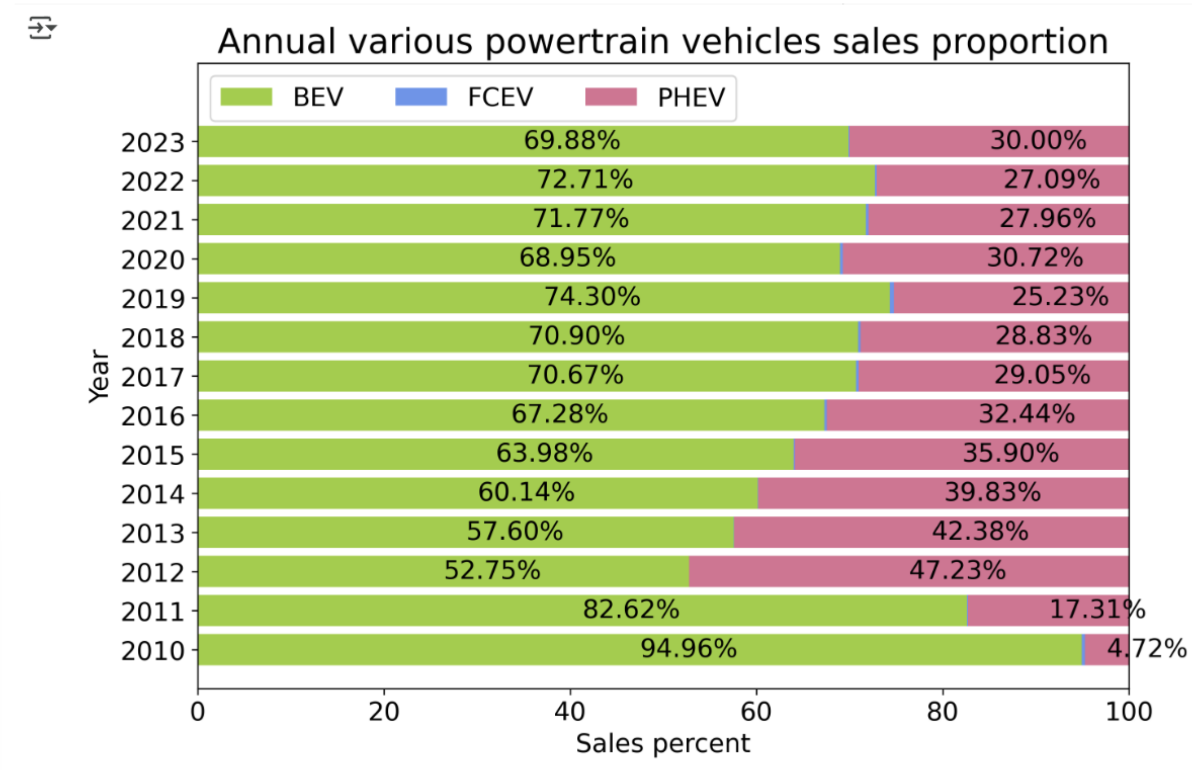

Figure 4.18 Chart about the proportional composition of sales of vehicles

This is the histogram of "Annual Percentage of Vehicle Sales by Powertrain", reflecting the sale percentages of three kinds of vehicles, namely BEVs, FCEVs, and PHEVs, from the period ranging between 2010 to 2023. Throughout these years from 2010 to 2023, BEVs were always at the top in the market. Though their market shares decreased, they have remained at a high level. The gradual growth in the market share of PHEVs demonstrates the interest and acceptance of hybrid technology in the market. FCEVs have not yet been able to capture a significant share of the market, reflecting challenges in their market acceptance.

```
[ ]  group_region=data_sale.groupby('region')['value'].sum().astype(int).reset_index()
     group_region.sort_values(by='value',ascending=False,inplace=True)
     group_region.reset_index(drop=True,inplace=True)
     new_row=pd.DataFrame([{'region':'rest  of  world','value':group_region['value'][10:].sum()}])
     group_region=group_region.drop(group_region[9:].index)
     group_region=pd.concat([group_region,new_row],ignore_index=True)
     print(group_region)
     fig,ax=plt.subplots(figsize=(6,6),dpi=300)
     pie,_,_=ax.pie(group_region['value'],labels=group_region['region'],autopct='%1.2f%%',pctdistance=0.8,radius=1,textpr
                    labeldistance=1.05,colors=plt.cm.Spectral(np.linspace(0,1,len(group_region['region']))))
     plt.setp(pie,width=0.5,edgecolor='k')
     ax.set_title('Proportion  of  total  sales  of  electric  vehicles',fontsize=20)
     plt.show()
```

Figure 4.19 Analyze the country with the highest annual sales of electric vehicles

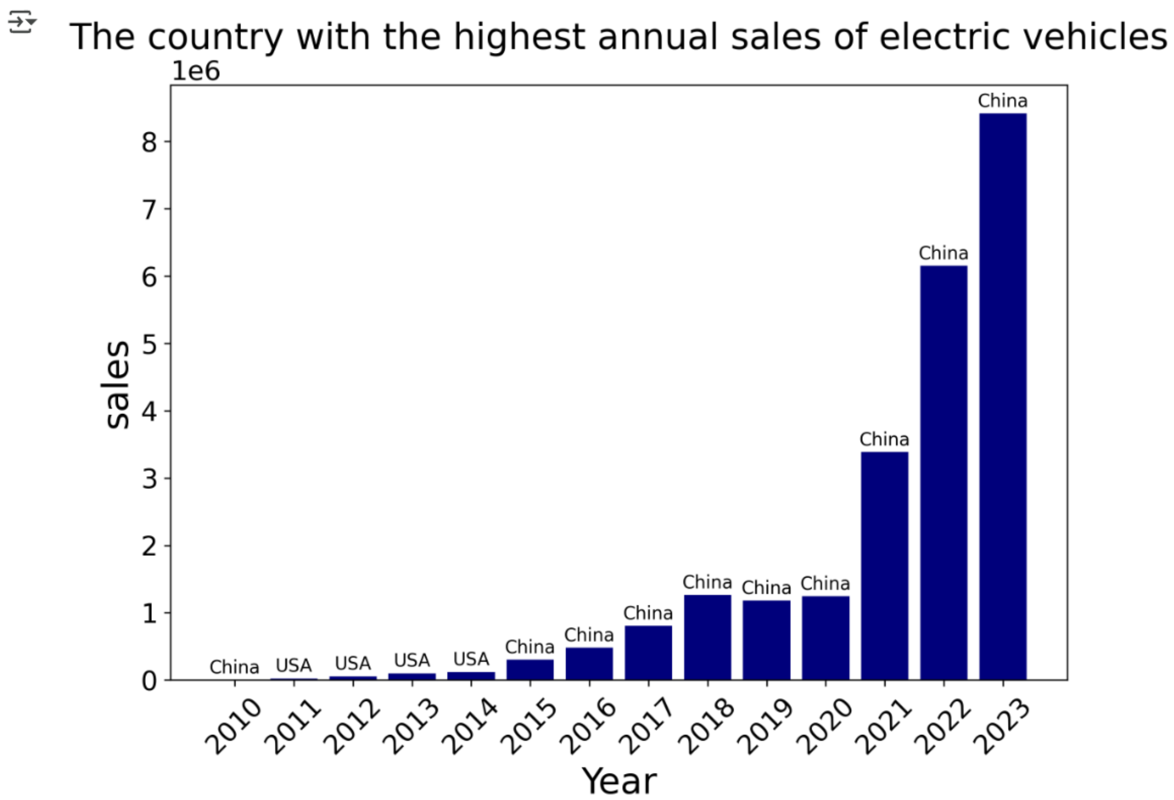

Figure 4.20 Chart about the trends of electric vehicles.

As can be seen from the above chart, except for the years 2012-2014, when the United States had the highest sales of electric vehicles, China has been the leading country in electric vehicle sales for the rest of the years. The reasons for this can be attributed to several factors:

1) China's vast population,

2) government policy support

3) the development of China's industrial technology.

```
[ ] group_region=data_sale.groupby('region')['value'].sum().astype(int).reset_index()
    group_region.sort_values(by='value',ascending=False,inplace=True)
    group_region.reset_index(drop=True,inplace=True)
    new_row=pd.DataFrame([{'region':'rest of world','value':group_region['value'][10:].sum()}])
    group_region=group_region.drop(group_region[9:].index)
    group_region=pd.concat([group_region,new_row],ignore_index=True)
    print(group_region)
    fig,ax=plt.subplots(figsize=(6,6),dpi=300)
    pie,_,_=ax.pie(group_region['value'],labels=group_region['region'],autopct='%1.2f%%',pctdistance=0.8,radius=1,textpr
                           labeldistance=1.05,colors=plt.cm.Spectral(np.linspace(0,1,len(group_region['region']))))
    plt.setp(pie,width=0.5,edgecolor='k')
    ax.set_title('Proportion of total sales of electric vehicles',fontsize=20)
    plt.show()
```

```
            region     value
0            China  23358308
1              USA   4770925
2          Germany   3012826
3           France   1666650
4   United Kingdom   1659853
5           Norway    879813
6      Netherlands    798918
7           Sweden    706166
8            Korea    676352
9    rest of world   4730788
```

Figure 4.21 Analyze the proportion of total sales of electric vehicles

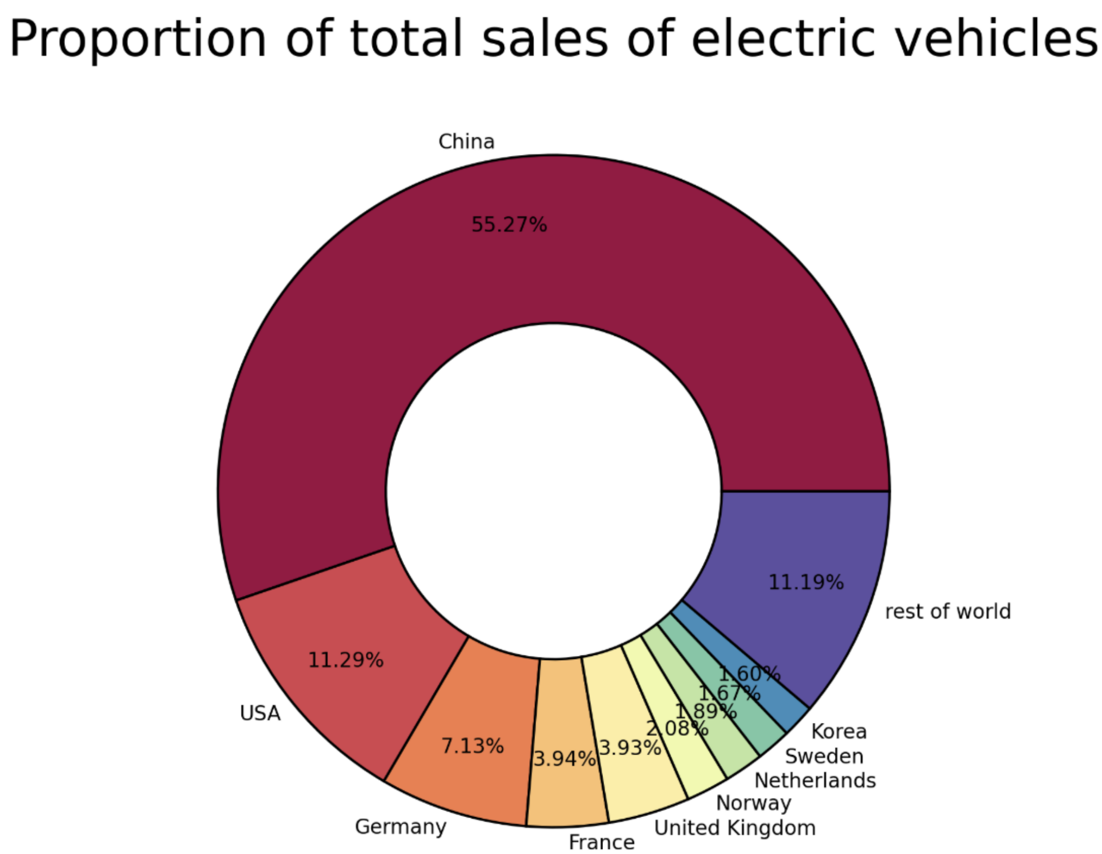

Figure 4.22 Chart about the proportion of total sales of electric vehicles

Among the total cumulative sales of electric vehicles over the years, China accounts for 55.27% of the market share, followed by the United States and Germany.