In [1]:

```
from IPython.display import Image
Image(url= "https://www.mbusa.com/content/dam/mb-nafta/us/eq/design/eqs580x4/interior/hotspots/MY23-EQS-SUV-TP-Hyperscreen-XL.j
```

Out[1]:

Required Modules

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import plotly
import plotly.figure_factory as ff
from plotly.offline import iplot
import plotly.express as px
import plotly.graph_objs as go
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.svm import SVR
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score,mean_squared_error
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import MinMaxScaler, StandardScaler
```

```python
from sklearn.model_selection import GridSearchCV
from math import sqrt
from tabulate import tabulate
from numpy import mean
from numpy import absolute
plotly.offline.init_notebook_mode()
%matplotlib inline
sns.set()
```

```
C:\Users\Mehul\anaconda3\Lib\site-packages\paramiko\transport.py:219: CryptographyDeprecationWarning: Blowfish has been depreca
ted
  "class": algorithms.Blowfish,
```

Data Description

In [3]:
```python
df_train = pd.read_csv(r'C:\Users\Mehul\Mercedes Task\train.csv')
df_test = pd.read_csv(r'C:\Users\Mehul\Mercedes Task\test.csv')
```

VIN (1-10) - The 1st 10 characters of each vehicle's Vehicle Identification Number (VIN).

County- The county in which the registered owner resides.

City - The city in which the registered owner resides.

State- The state in which the registered owner resides.

ZIP Code - The 5-digit zip code in which the registered owner resides.

Model Year - The model year of the vehicle is determined by decoding the Vehicle Identification Number (VIN).

Make- The manufacturer of the vehicle, determined by decoding the Vehicle Identification Number (VIN).

Model- The model of the vehicle is determined by decoding the Vehicle Identification Number (VIN).

Electric Vehicle Type - This distinguishes the vehicle as all-electric or a plug-in hybrid.

Clean Alternative Fuel Vehicle (CAFV) Eligibility - This categorizes vehicles as Clean Alternative Fuel Vehicles (CAFVs) based on the fuel requirement and electric-only range requirement.

Electric Range - Describes how far a vehicle can travel purely on its electric charge.

Base MSRP - This is the lowest Manufacturer's Suggested Retail Price (MSRP) for any trim level of the model in question.

Legislative District - The specific section of Washington State that the vehicle's owner resides in, as represented in the state legislature.

DOL Vehicle ID - Unique number assigned to each vehicle by the Department of Licensing for identification purposes.

Vehicle Location - The center of the ZIP Code for the registered vehicle.

Electric Utility - This is the electric power retail service territory serving the address of the registered vehicle.

Expected Price - This is the expected price of the vehicle.

Data Overview

In [4]:
```
df_train.head()
```

Out[4]:

	ID	VIN (1-10)	County	City	State	ZIP Code	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP	Legislative District	DOL Vehicle ID	V Lo
0	EV67734	5YJSA1E29J	Clallam	SEQUIM	WA	98382.0	2018.0	TESLA	MODEL S	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	249	0	24.0	474843043	P (-123.0 48.05
1	EV41866	JTDKARFPXL	Island	OAK HARBOR	WA	98277.0	2020.0	TOYOTA	PRIUS PRIME	Plug-in Hybrid Electric Vehicle (PHEV)	Not eligible due to low battery range	25	0	10.0	112793138	P (-122. 48.31

2	EV66838	7SAYGDEF2N	King	KENT	WA	98030.0	2022.0	TESLA	MODEL Y	Battery Electric Vehicle (BEV)	Eligibility unknown as battery range has not b...	0	0	47.0	187580606	(-122.1 47.36
3	EV66343	7SAYGDEF0N	King	RENTON	WA	98059.0	2022.0	TESLA	MODEL Y	Battery Electric Vehicle (BEV)	Eligibility unknown as battery range has not b...	0	0	5.0	190762148	(-122.1 47.49
4	EV88468	JHMZC5F3XJ	Snohomish	LAKE STEVENS	WA	98258.0	2018.0	HONDA	CLARITY	Plug-in Hybrid Electric Vehicle (PHEV)	Clean Alternative Fuel Vehicle Eligible	47	0	39.0	227929733	(-122.0 48.01

```
In [5]: df_test.head()
```

Out[5]:

	ID	VIN (1-10)	County	City	State	ZIP Code	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP	Legislative District	DOL Vehicle ID	
0	EV45181	1G1FW6S06N	King	ENUMCLAW	WA	98022.0	2022.0	CHEVROLET	BOLT EV	Battery Electric Vehicle (BEV)	Eligibility unknown as battery range has not b...	0	0	31.0	166559368	(
1	EV69055	WVWKR7AU5K	King	SAMMAMISH	WA	98075.0	2019.0	VOLKSWAGEN	E-GOLF	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	125	0	41.0	3595182	(
2	EV60667	WDDVP9AB5H	King	TUKWILA	WA	98188.0	2017.0	MERCEDES-BENZ	B-CLASS	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	87	0	11.0	180632266	(
3	EV89394	5YJSA1CG1D	King	SEATTLE	WA	98178.0	2013.0	TESLA	MODEL S	Battery Electric Vehicle (BEV)	Clean Alternative Fuel	208	69900	37.0	8032475	(

											Vehicle Eligible				
4	EV79353	5YJ3E1EB5L	King	SEATTLE	WA	98121.0	2020.0	TESLA	MODEL 3	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	322	0	43.0	112536070

In [6]:
```python
print('Training data shape',df_train.shape)
print('Test data shape',df_test.shape)
```

```
Training data shape (51482, 18)
Test data shape (12871, 18)
```

In [7]:
```python
print(f"Duplicates in Train Dataset is:{df_train.duplicated().sum()},({100*df_train.duplicated().sum()/len(df_train)})%")
print(f"Duplicates in Test Dataset is:{df_test.duplicated().sum()},({100*df_test.duplicated().sum()/len(df_test)})%")
```

```
Duplicates in Train Dataset is:0,(0.0)%
Duplicates in Test Dataset is:0,(0.0)%
```

In [8]:
```python
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51482 entries, 0 to 51481
Data columns (total 18 columns):
 #   Column                                             Non-Null Count  Dtype  
---  ------                                             --------------  -----  
 0   ID                                                 51482 non-null  object 
 1   VIN (1-10)                                         51482 non-null  object 
 2   County                                             51479 non-null  object 
 3   City                                               51475 non-null  object 
 4   State                                              51474 non-null  object 
 5   ZIP Code                                           51476 non-null  float64
 6   Model Year                                         51476 non-null  float64
 7   Make                                               51479 non-null  object 
 8   Model                                              51473 non-null  object 
 9   Electric Vehicle Type                              51482 non-null  object 
 10  Clean Alternative Fuel Vehicle (CAFV) Eligibility  51482 non-null  object 
 11  Electric Range                                     51482 non-null  int64  
 12  Base MSRP                                          51482 non-null  int64  
 13  Legislative District                               51337 non-null  float64
 14  DOL Vehicle ID                                     51482 non-null  int64  
 15  Vehicle Location                                   51088 non-null  object 
 16  Electric Utility                                   50908 non-null  object 
 17  Expected Price ($1k)                               51482 non-null  object 
```

```
dtypes: float64(3), int64(3), object(12)
memory usage: 7.1+ MB
```

In [9]:
```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12871 entries, 0 to 12870
Data columns (total 18 columns):
 #   Column                                             Non-Null Count  Dtype  
---  ------                                             --------------  -----  
 0   ID                                                 12871 non-null  object 
 1   VIN (1-10)                                         12871 non-null  object 
 2   County                                             12870 non-null  object 
 3   City                                               12869 non-null  object 
 4   State                                              12868 non-null  object 
 5   ZIP Code                                           12871 non-null  float64
 6   Model Year                                         12870 non-null  float64
 7   Make                                               12870 non-null  object 
 8   Model                                              12867 non-null  object 
 9   Electric Vehicle Type                              12871 non-null  object 
 10  Clean Alternative Fuel Vehicle (CAFV) Eligibility  12871 non-null  object 
 11  Electric Range                                     12871 non-null  int64  
 12  Base MSRP                                          12871 non-null  int64  
 13  Legislative District                               12847 non-null  float64
 14  DOL Vehicle ID                                     12871 non-null  int64  
 15  Vehicle Location                                   12755 non-null  object 
 16  Electric Utility                                   12723 non-null  object 
 17  Expected Price ($1k)                               12871 non-null  object 
dtypes: float64(3), int64(3), object(12)
memory usage: 1.8+ MB
```

In [10]:
```
df_train.describe()
```

Out[10]:

	ZIP Code	Model Year	Electric Range	Base MSRP	Legislative District	DOL Vehicle ID
count	51476.000000	51476.000000	51482.000000	51482.000000	51337.000000	5.148200e+04
mean	98130.866734	2018.177714	106.757605	2509.410085	29.961918	1.973874e+08
std	3005.260280	2.729389	103.955854	12397.643780	14.658074	1.068871e+08
min	745.000000	1993.000000	0.000000	0.000000	0.000000	4.385000e+03
25%	98052.000000	2017.000000	14.000000	0.000000	19.000000	1.372688e+08
50%	98119.000000	2018.000000	73.000000	0.000000	34.000000	1.753387e+08
75%	98370.000000	2021.000000	215.000000	0.000000	43.000000	2.300245e+08
max	99567.000000	2022.000000	337.000000	845000.000000	49.000000	4.789346e+08

In [11]:
```
df_test.describe()
```

Out[11]:

	ZIP Code	Model Year	Electric Range	Base MSRP	Legislative District	DOL Vehicle ID
count	12871.000000	12870.000000	12871.000000	12871.000000	12847.000000	1.287100e+04
mean	98193.789682	2018.220202	107.714474	2587.311009	29.911886	1.969029e+08
std	2158.000424	2.715968	104.644890	12424.164249	14.673806	1.071877e+08
min	9751.000000	1998.000000	0.000000	0.000000	0.000000	1.205000e+04
25%	98052.000000	2017.000000	13.000000	0.000000	19.000000	1.373641e+08
50%	98121.000000	2019.000000	75.000000	0.000000	34.000000	1.755213e+08
75%	98370.000000	2021.000000	215.000000	0.000000	43.000000	2.290691e+08
max	99701.000000	2022.000000	337.000000	184400.000000	49.000000	4.789263e+08

In [12]:
```
df1 = (df_train.isnull().sum()[df_train.isnull().sum()>0]).to_frame().rename(columns={0:"Number of Missing values"})
df1["% of Missing Values"] = round((100*df_train.isnull().sum()[df_train.isnull().sum()>0]/len(df_train)),3)
df1
```

Out[12]:

	Number of Missing values	% of Missing Values
County	3	0.006
City	7	0.014
State	8	0.016
ZIP Code	6	0.012
Model Year	6	0.012
Make	3	0.006
Model	9	0.017
Legislative District	145	0.282
Vehicle Location	394	0.765
Electric Utility	574	1.115

In [13]:
```
df1 = (df_test.isnull().sum()[df_test.isnull().sum()>0]).to_frame().rename(columns={0:"Number of Missing values"})
df1["% of Missing Values"] = round((100*df_test.isnull().sum()[df_test.isnull().sum()>0]/len(df_test)),3)
df1
```

Out[13]:

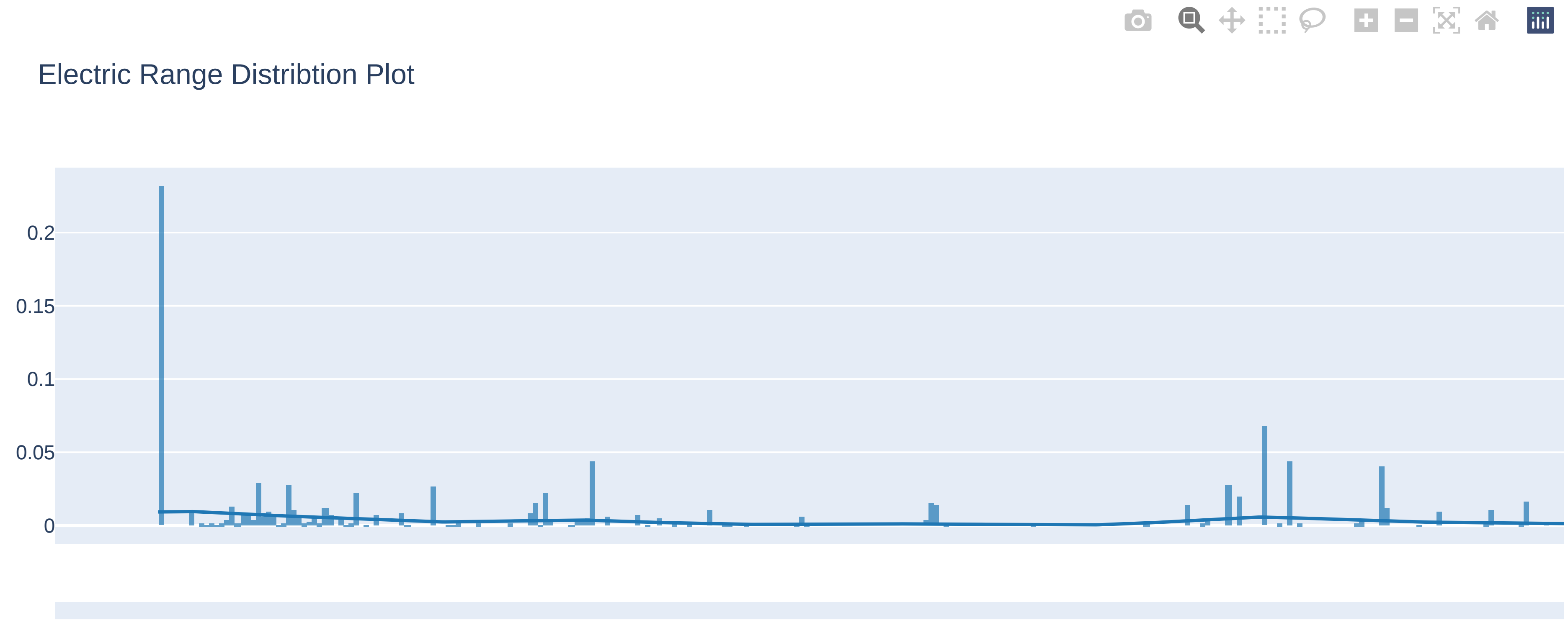
	Number of Missing values	% of Missing Values
County	1	0.008

City	2	0.016
State	3	0.023
Model Year	1	0.008
Make	1	0.008
Model	4	0.031
Legislative District	24	0.186
Vehicle Location	116	0.901
Electric Utility	148	1.150

Exploratory Data Analysis

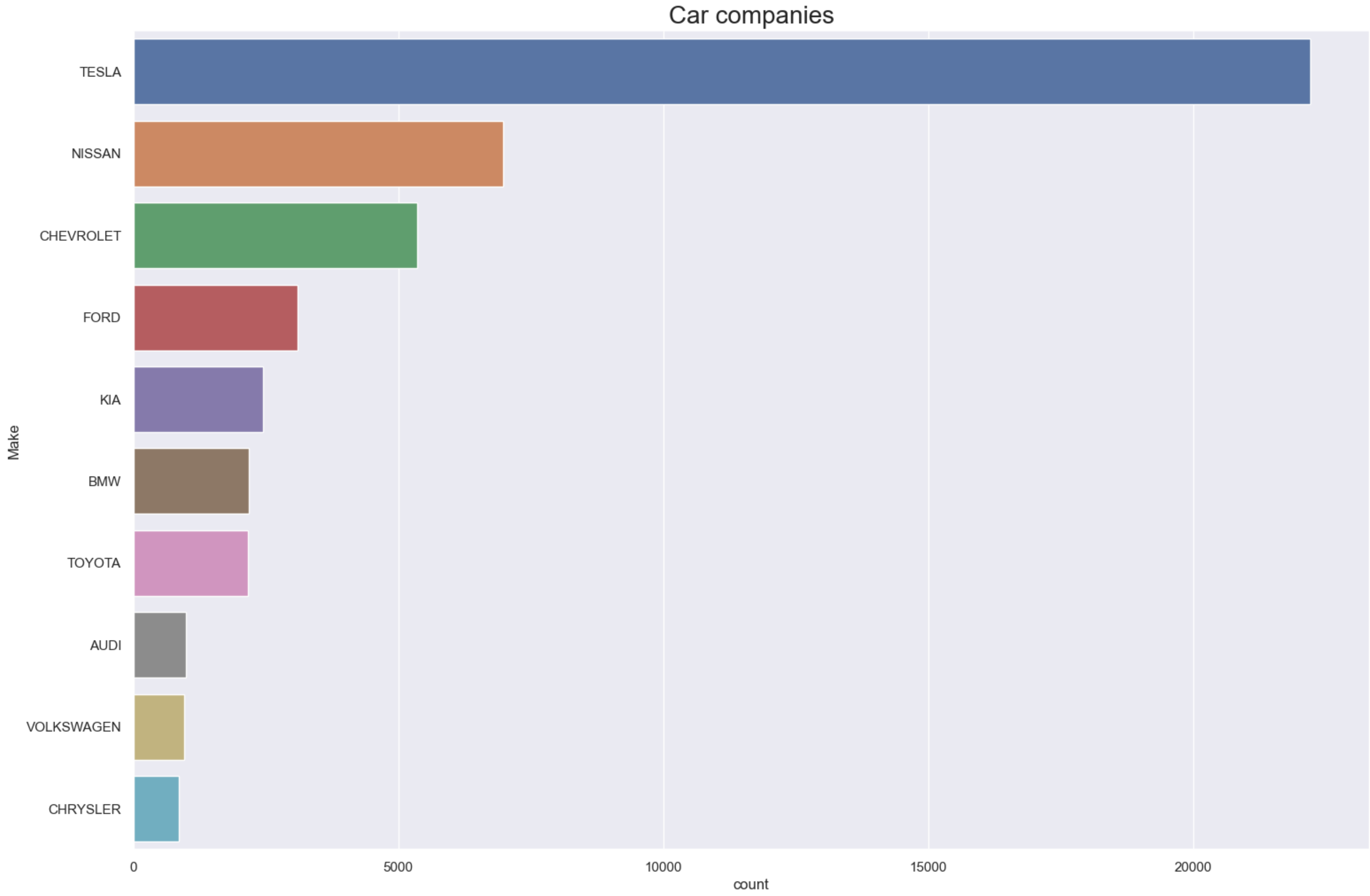
In [14]:

```
fig=ff.create_distplot([df_train['Electric Range']],['Electric Range'])
fig.update_layout(title_text='Electric Range Distribtion Plot')
fig.show();
```

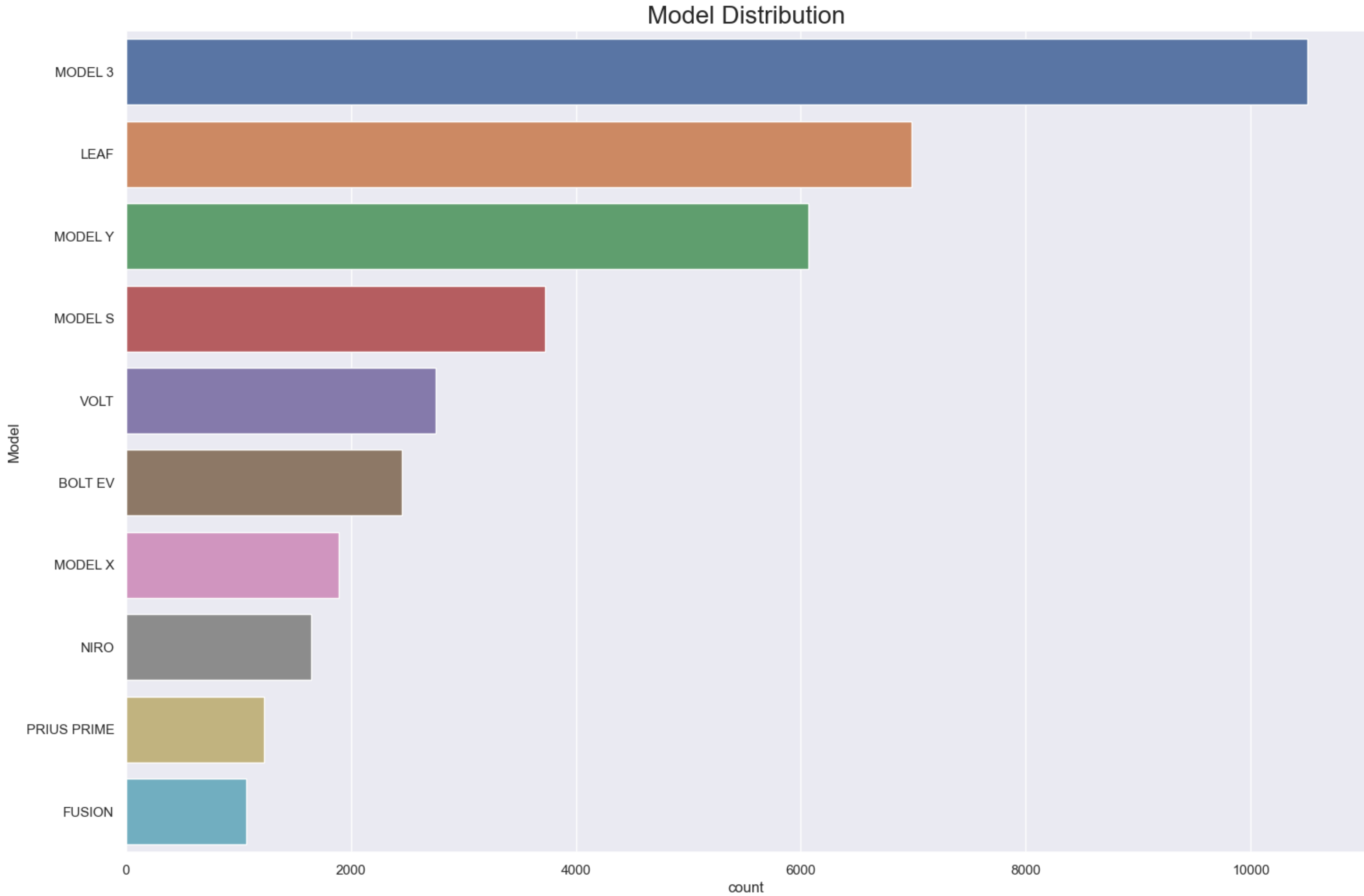


In [15]:

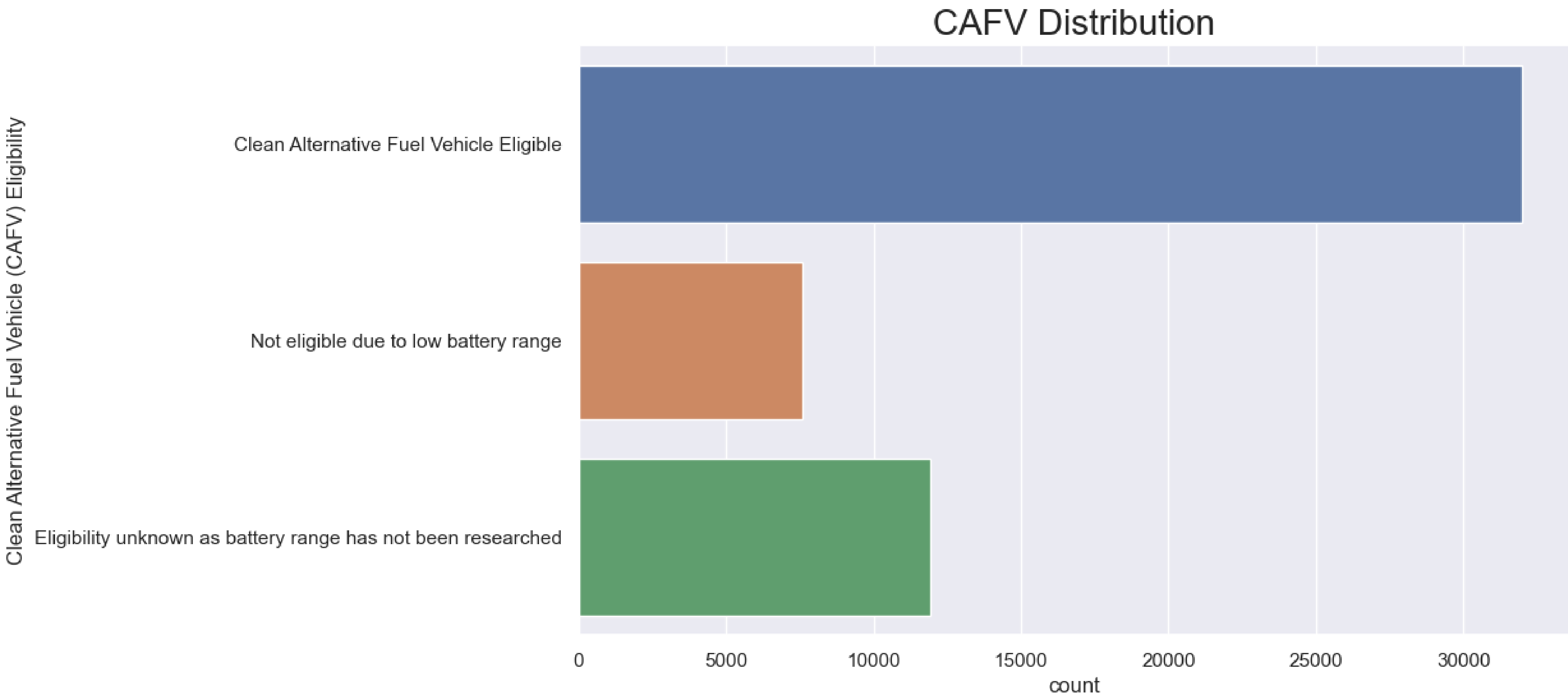
```
plt.figure(figsize = (18, 12))
sns.countplot(y = df_train['Make'], order=df_train.Make.value_counts().iloc[:10].index)
plt.title("Car companies", fontsize = 20)
plt.show()
```



In [16]:
```
plt.figure(figsize = (18, 12))
sns.countplot(y = df_train['Model'],order=df_train.Model.value_counts().iloc[:10].index)
plt.title("Model Distribution", fontsize = 20)
plt.show()
```

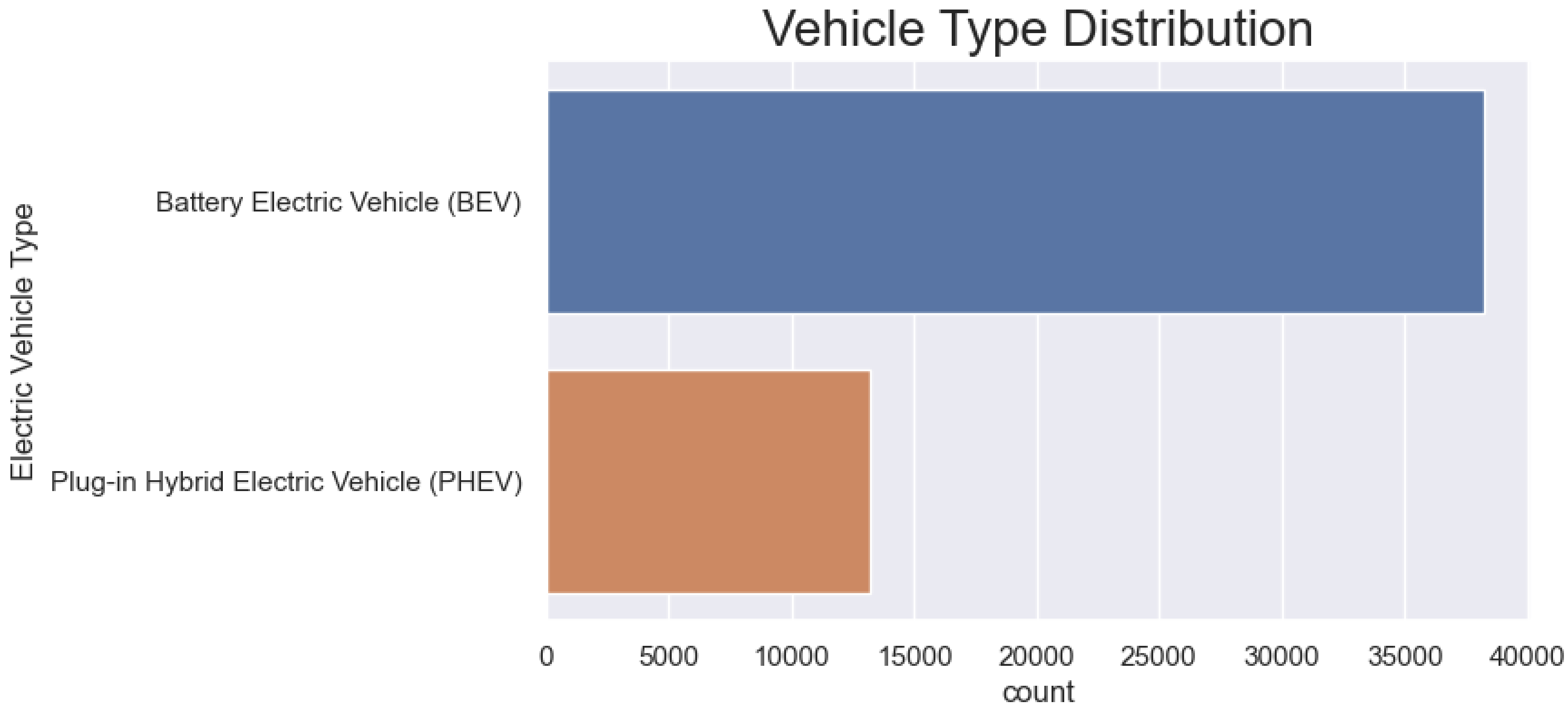


In [17]:
```
plt.figure(figsize = (10, 6))
sns.countplot(y = df_train['Clean Alternative Fuel Vehicle (CAFV) Eligibility'])
plt.title("CAFV Distribution", fontsize = 20)
plt.show()
```



In [18]:

```
plt.figure(figsize = (7, 4))
sns.countplot(y = df_train['Electric Vehicle Type'])
plt.title("Vehicle Type Distribution", fontsize = 20)
plt.show()
```



Data Preprocessing

In [19]:
```
df_train['State'].value_counts()
```

Out[19]:
```
WA    51325
CA       32
MD       20
VA       19
TX       10
OR        5
FL        4
NC        4
GA        4
IL        4
NV        4
PA        4
TN        3
HI        3
CT        3
NE        3
AP        3
```

```
MA        2
NJ        2
KS        2
DC        2
MT        2
NY        2
DE        1
MO        1
RI        1
NM        1
PR        1
IN        1
LA        1
WI        1
AZ        1
MS        1
WY        1
UT        1
Name: State, dtype: int64
```

In [20]:
```
df_test['State'].value_counts()
```

Out[20]:
```
WA    12843
CA        8
MD        2
VA        2
DC        1
OR        1
NM        1
TX        1
NV        1
NC        1
FL        1
AE        1
SD        1
AK        1
IN        1
HI        1
NY        1
Name: State, dtype: int64
```

In [21]:
```
df_train.drop('State',axis=1,inplace=True)
df_test.drop('State',axis=1,inplace=True)
```

In [22]:
```
df_train.drop(columns = ['City', 'County', 'ZIP Code',
                                       'ID', 'VIN (1-10)', 'Vehicle Location', 'DOL Vehicle ID',
                                       'Electric Utility'],inplace=True)
```

```python
df_test.drop(columns = ['City', 'County', 'ZIP Code',
                        'ID', 'VIN (1-10)', 'Vehicle Location', 'DOL Vehicle ID',
                        'Electric Utility'],inplace=True)
```

In [23]:
```python
df_train
```

Out[23]:

	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP	Legislative District	Expected Price ($1k)
0	2018.0	TESLA	MODEL S	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	249	0	24.0	69
1	2020.0	TOYOTA	PRIUS PRIME	Plug-in Hybrid Electric Vehicle (PHEV)	Not eligible due to low battery range	25	0	10.0	40
2	2022.0	TESLA	MODEL Y	Battery Electric Vehicle (BEV)	Eligibility unknown as battery range has not b...	0	0	47.0	78
3	2022.0	TESLA	MODEL Y	Battery Electric Vehicle (BEV)	Eligibility unknown as battery range has not b...	0	0	5.0	78
4	2018.0	HONDA	CLARITY	Plug-in Hybrid Electric Vehicle (PHEV)	Clean Alternative Fuel Vehicle Eligible	47	0	39.0	24.283
...	...	...	...	...	...	...	...	...	...
51477	2016.0	NISSAN	LEAF	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	84	0	46.0	27
51478	2019.0	NISSAN	LEAF	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	150	0	2.0	35
51479	2016.0	BMW	I3	Plug-in Hybrid Electric Vehicle (PHEV)	Clean Alternative Fuel Vehicle Eligible	72	0	39.0	19
51480	2019.0	TESLA	MODEL 3	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	220	0	45.0	57
51481	2018.0	TESLA	MODEL S	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	249	0	41.0	69

51482 rows × 9 columns

In [24]:
```python
def target_drop(df):
    l=df[(df['Expected Price ($1k)']=='N/')].index
    for i in l:
        df.drop(i,axis=0,inplace=True)
    df['Expected Price ($1k)'] = pd.to_numeric(df['Expected Price ($1k)'], downcast='float')
    df['Expected Price ($1k)'] = df['Expected Price ($1k)']*1000
    df.rename(columns = {'Expected Price ($1k)':'Expected Price'},inplace=True)
    return df
```

In [25]:
```
target_drop(df_train)
```

Out[25]:

	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP	Legislative District	Expected Price
0	2018.0	TESLA	MODEL S	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	249	0	24.0	69000.0
1	2020.0	TOYOTA	PRIUS PRIME	Plug-in Hybrid Electric Vehicle (PHEV)	Not eligible due to low battery range	25	0	10.0	40000.0
2	2022.0	TESLA	MODEL Y	Battery Electric Vehicle (BEV)	Eligibility unknown as battery range has not b...	0	0	47.0	78000.0
3	2022.0	TESLA	MODEL Y	Battery Electric Vehicle (BEV)	Eligibility unknown as battery range has not b...	0	0	5.0	78000.0
4	2018.0	HONDA	CLARITY	Plug-in Hybrid Electric Vehicle (PHEV)	Clean Alternative Fuel Vehicle Eligible	47	0	39.0	24283.0
...	...	...	...	...	...	...	...	...	...
51477	2016.0	NISSAN	LEAF	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	84	0	46.0	27000.0
51478	2019.0	NISSAN	LEAF	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	150	0	2.0	35000.0
51479	2016.0	BMW	I3	Plug-in Hybrid Electric Vehicle (PHEV)	Clean Alternative Fuel Vehicle Eligible	72	0	39.0	19000.0
51480	2019.0	TESLA	MODEL 3	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	220	0	45.0	57000.0
51481	2018.0	TESLA	MODEL S	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	249	0	41.0	69000.0

51473 rows × 9 columns

In [26]:
```
target_drop(df_test)
```

Out[26]:

	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP	Legislative District	Expected Price
0	2022.0	CHEVROLET	BOLT EV	Battery Electric Vehicle (BEV)	Eligibility unknown as battery range has not b...	0	0	31.0	33500.0
1	2019.0	VOLKSWAGEN	E-GOLF	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	125	0	41.0	22200.0
2	2017.0	MERCEDES-BENZ	B-CLASS	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	87	0	11.0	36000.0

3	2013.0	TESLA	MODEL S	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	208	69900	37.0	33890.0
4	2020.0	TESLA	MODEL 3	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	322	0	43.0	50000.0
...	...	...	...	...	...	...	...	...	...
12866	2020.0	TESLA	MODEL X	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	289	0	49.0	102000.0
12867	2017.0	CHEVROLET	BOLT EV	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	238	0	48.0	20000.0
12868	2017.0	TESLA	MODEL S	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	210	0	17.0	60000.0
12869	2013.0	NISSAN	LEAF	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	75	0	46.0	18000.0
12870	2018.0	TESLA	MODEL 3	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	215	0	17.0	69000.0

12867 rows × 9 columns

In [27]:
```
df1 = (df_train.isnull().sum()[df_train.isnull().sum()>0]).to_frame().rename(columns={0:"Number of Missing values"})
df1["% of Missing Values"] = round((100*df_train.isnull().sum()[df_train.isnull().sum()>0]/len(df_train)),3)
df1
```

Out[27]:

	Number of Missing values	% of Missing Values
Model Year	6	0.012
Make	3	0.006
Legislative District	145	0.282

In [28]:
```
df1 = (df_test.isnull().sum()[df_test.isnull().sum()>0]).to_frame().rename(columns={0:"Number of Missing values"})
df1["% of Missing Values"] = round((100*df_test.isnull().sum()[df_test.isnull().sum()>0]/len(df_test)),3)
df1
```

Out[28]:

	Number of Missing values	% of Missing Values
Model Year	1	0.008
Make	1	0.008
Legislative District	24	0.187

In [29]:
```
df_train = df_train[df_train['Model Year'].notna()]
df_train = df_train['Make'].notna()]
```

```
df_test = df_test[df_test['Model Year'].notna()]
df_test = df_test[df_test['Make'].notna()]
```

In [30]:
```
df_train.fillna(df_train['Legislative District'].value_counts().index[0],inplace=True)
df_test.fillna(df_test['Legislative District'].value_counts().index[0],inplace=True)
```

In [31]:
```
df_train.isnull().sum()
```

Out[31]:
```
Model Year                                           0
Make                                                 0
Model                                                0
Electric Vehicle Type                                0
Clean Alternative Fuel Vehicle (CAFV) Eligibility    0
Electric Range                                       0
Base MSRP                                            0
Legislative District                                 0
Expected Price                                       0
dtype: int64
```

In [32]:
```
df_test.isnull().sum()
```

Out[32]:
```
Model Year                                           0
Make                                                 0
Model                                                0
Electric Vehicle Type                                0
Clean Alternative Fuel Vehicle (CAFV) Eligibility    0
Electric Range                                       0
Base MSRP                                            0
Legislative District                                 0
Expected Price                                       0
dtype: int64
```

Feature Engineering

In [33]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51464 entries, 0 to 51481
Data columns (total 9 columns):
 #   Column                                             Non-Null Count  Dtype  
---  ------                                             --------------  -----  
 0   Model Year                                         51464 non-null  float64
 1   Make                                               51464 non-null  object 
 2   Model                                              51464 non-null  object 
 3   Electric Vehicle Type                              51464 non-null  object 
 4   Clean Alternative Fuel Vehicle (CAFV) Eligibility  51464 non-null  object 
 5   Electric Range                                     51464 non-null  int64  
```

```
 6   Base MSRP                                         51464 non-null  int64
 7   Legislative District                              51464 non-null  float64
 8   Expected Price                                    51464 non-null  float32
dtypes: float32(1), float64(2), int64(2), object(4)
memory usage: 3.7+ MB
```

In [34]:
```python
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12865 entries, 0 to 12870
Data columns (total 9 columns):
 #   Column                                            Non-Null Count  Dtype
---  ------                                            --------------  -----
 0   Model Year                                        12865 non-null  float64
 1   Make                                              12865 non-null  object
 2   Model                                             12865 non-null  object
 3   Electric Vehicle Type                             12865 non-null  object
 4   Clean Alternative Fuel Vehicle (CAFV) Eligibility 12865 non-null  object
 5   Electric Range                                    12865 non-null  int64
 6   Base MSRP                                         12865 non-null  int64
 7   Legislative District                              12865 non-null  float64
 8   Expected Price                                    12865 non-null  float32
dtypes: float32(1), float64(2), int64(2), object(4)
memory usage: 954.8+ KB
```

In [35]:
```python
df_train.groupby('Make')['Model']
for i in df_train.groupby('Make')['Model']:
    df_train.replace(list(i[1].unique()),[m+1 for m in range(len(list(i[1].unique())))],inplace=True)
```

In [36]:
```python
df_train
```

Out[36]:

	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP	Legislative District	Expected Price
0	2018.0	TESLA	1	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	249	0	24.0	69000.0
1	2020.0	TOYOTA	1	Plug-in Hybrid Electric Vehicle (PHEV)	Not eligible due to low battery range	25	0	10.0	40000.0
2	2022.0	TESLA	2	Battery Electric Vehicle (BEV)	Eligibility unknown as battery range has not b...	0	0	47.0	78000.0
3	2022.0	TESLA	2	Battery Electric Vehicle (BEV)	Eligibility unknown as battery range has not b...	0	0	5.0	78000.0
4	2018.0	HONDA	1	Plug-in Hybrid Electric Vehicle (PHEV)	Clean Alternative Fuel Vehicle Eligible	47	0	39.0	24283.0
...	...	...	...	...	...	...	...	...	...
51477	2016.0	NISSAN	1	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	84	0	46.0	27000.0

51478	2019.0	NISSAN	1	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	150	0	2.0	35000.0
51479	2016.0	BMW	1	Plug-in Hybrid Electric Vehicle (PHEV)	Clean Alternative Fuel Vehicle Eligible	72	0	39.0	19000.0
51480	2019.0	TESLA	3	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	220	0	45.0	57000.0
51481	2018.0	TESLA	1	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	249	0	41.0	69000.0

51464 rows × 9 columns

In [37]:

```
df_test.groupby('Make')['Model']
for i in df_test.groupby('Make')['Model']:
    df_test.replace(list(i[1].unique()),[m+1 for m in range(len(list(i[1].unique())))],inplace=True)
```

In [38]:

```
df_test
```

Out[38]:

	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP	Legislative District	Expected Price
0	2022.0	CHEVROLET	1	Battery Electric Vehicle (BEV)	Eligibility unknown as battery range has not b...	0	0	31.0	33500.0
1	2019.0	VOLKSWAGEN	1	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	125	0	41.0	22200.0
2	2017.0	MERCEDES-BENZ	1	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	87	0	11.0	36000.0
3	2013.0	TESLA	1	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	208	69900	37.0	33890.0
4	2020.0	TESLA	2	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	322	0	43.0	50000.0
...	...	...	...	...	...	...	...	...	...
12866	2020.0	TESLA	4	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	289	0	49.0	102000.0
12867	2017.0	CHEVROLET	1	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	238	0	48.0	20000.0
12868	2017.0	TESLA	1	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	210	0	17.0	60000.0
12869	2013.0	NISSAN	1	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	75	0	46.0	18000.0
12870	2018.0	TESLA	2	Battery Electric Vehicle (BEV)	Clean Alternative Fuel Vehicle Eligible	215	0	17.0	69000.0

12865 rows × 9 columns

In [39]:

```
df_train['Electric Vehicle Type'].replace(df_train['Electric Vehicle Type'].unique(),\
                                          [m+1 for m in range(len(df_train['Electric Vehicle Type'].unique()))],inplace=True)
df_train['Clean Alternative Fuel Vehicle (CAFV) Eligibility'].replace(df_train['Clean Alternative Fuel Vehicle (CAFV) Eligibili
                                          [m+1 for m in range(len(df_train['Clean Alternative Fuel Vehicle (CAFV) Eligibility']
df_train['Make'].replace(df_train['Make'].unique(),[m+1 for m in range(len(df_train['Make'].unique()))],inplace=True)
```

In [40]:

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 51464 entries, 0 to 51481
Data columns (total 9 columns):
 #   Column                                             Non-Null Count  Dtype
---  ------                                             --------------  -----
 0   Model Year                                         51464 non-null  float64
 1   Make                                               51464 non-null  int64
```

```
 2   Model                                         51464 non-null  int64
 3   Electric Vehicle Type                         51464 non-null  int64
 4   Clean Alternative Fuel Vehicle (CAFV) Eligibility  51464 non-null  int64
 5   Electric Range                                51464 non-null  int64
 6   Base MSRP                                     51464 non-null  int64
 7   Legislative District                          51464 non-null  float64
 8   Expected Price                                51464 non-null  float32
dtypes: float32(1), float64(2), int64(6)
memory usage: 3.7 MB
```

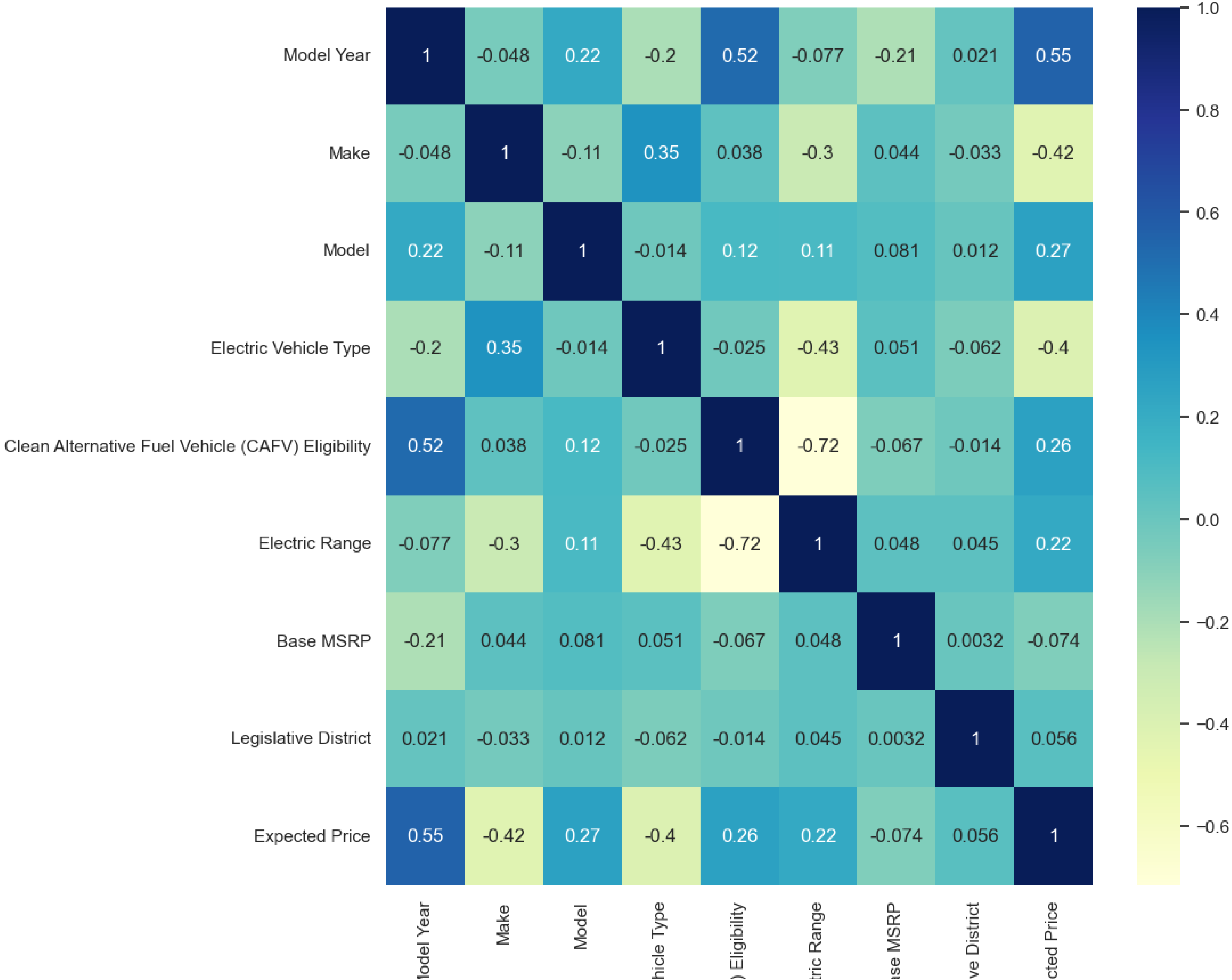
In [41]:
```
df_test['Electric Vehicle Type'].replace(df_test['Electric Vehicle Type'].unique(),\
                                          [m+1 for m in range(len(df_test['Electric Vehicle Type'].unique()))],inplace=True)
df_test['Clean Alternative Fuel Vehicle (CAFV) Eligibility'].replace(df_test['Clean Alternative Fuel Vehicle (CAFV) Eligibility
                                          [m+1 for m in range(len(df_test['Clean Alternative Fuel Vehicle (CAFV) Eligibility'].u
df_test['Make'].replace(df_test['Make'].unique(),[m+1 for m in range(len(df_test['Make'].unique()))],inplace=True)
```

In [42]:
```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 12865 entries, 0 to 12870
Data columns (total 9 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   Model Year                                    12865 non-null  float64
 1   Make                                          12865 non-null  int64
 2   Model                                         12865 non-null  int64
 3   Electric Vehicle Type                         12865 non-null  int64
 4   Clean Alternative Fuel Vehicle (CAFV) Eligibility  12865 non-null  int64
 5   Electric Range                                12865 non-null  int64
 6   Base MSRP                                     12865 non-null  int64
 7   Legislative District                          12865 non-null  float64
 8   Expected Price                                12865 non-null  float32
dtypes: float32(1), float64(2), int64(6)
memory usage: 954.8 KB
```

In [43]:
```
plt.figure(figsize = (10, 10))
sns.heatmap(df_train.corr(), cmap="YlGnBu", annot=True)
```

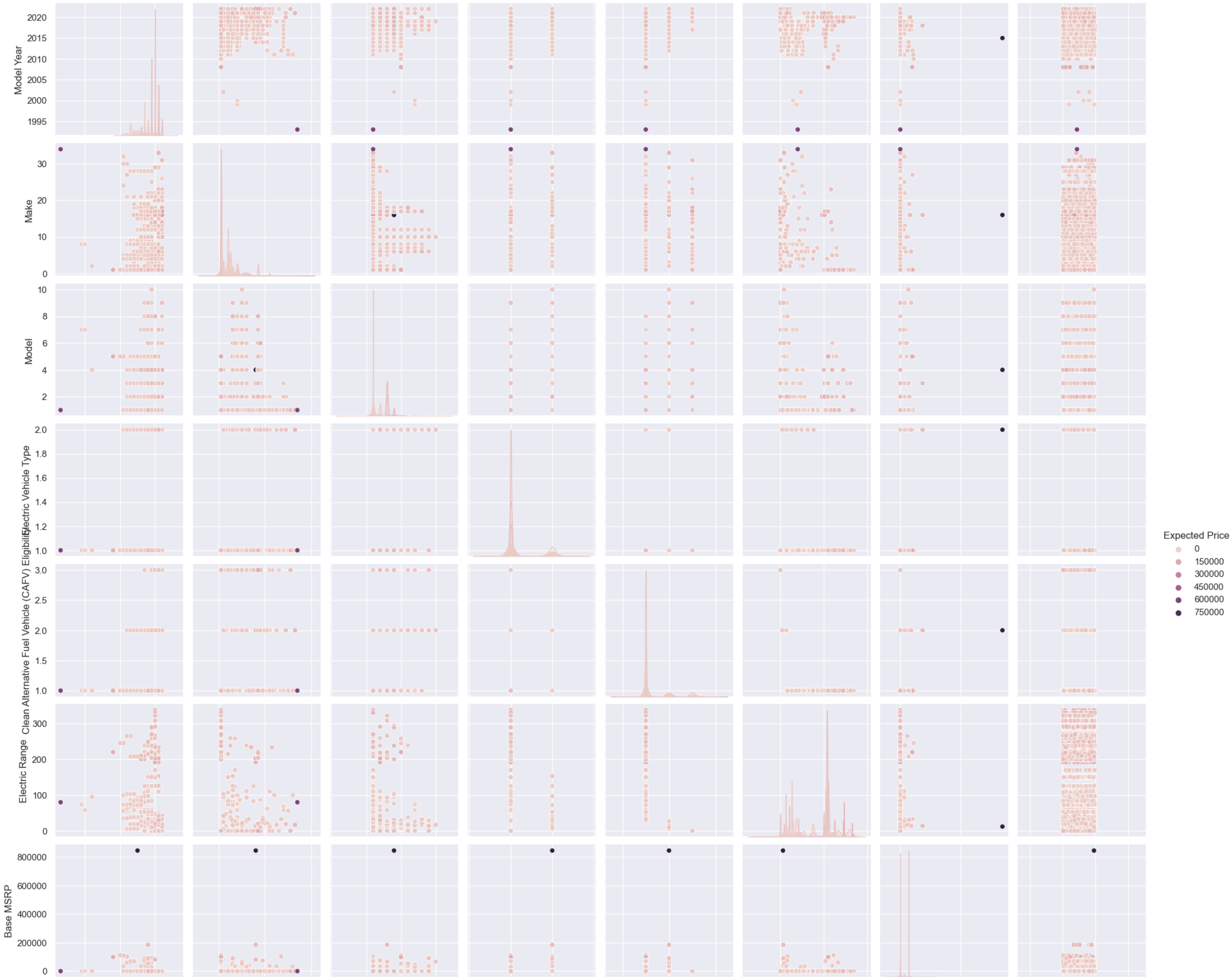
Out[43]:
```
<Axes: >
```


Model Year
Make
Model
Electric Vehicle Type
Clean Alternative Fuel Vehicle (CAFV) Eligibility
Electric Range
Base MSRP
Legislative District
Expected Price
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

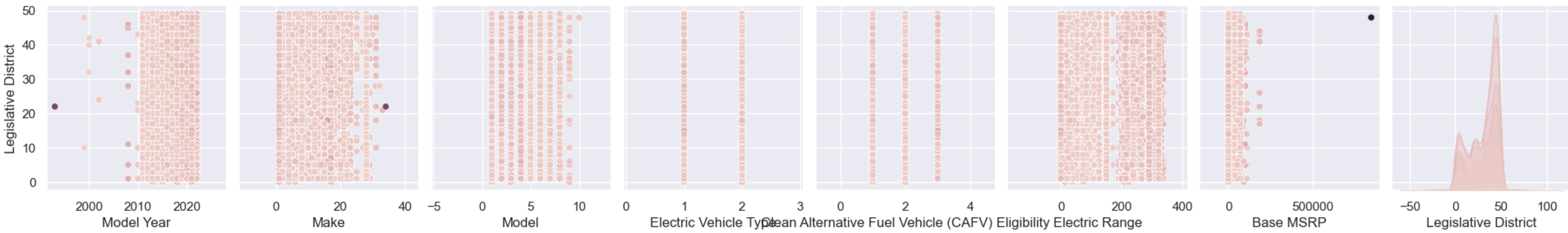
In [44]:
```
plt.figure(figsize = (20, 20))
sns.pairplot(df_train, hue ='Expected Price')
```

Out[44]:
```
<seaborn.axisgrid.PairGrid at 0x1626536c9d0>

<Figure size 2000x2000 with 0 Axes>
```



Model Year
Make
Model
Electric Vehicle Type
Clean Alternative Fuel Vehicle (CAFV) Eligibility
Electric Range
Base MSRP
Expected Price
0
150000
300000
450000
600000
750000



Data Normalization

In [45]:
```
scaler = StandardScaler()
df_train_scaled = pd.DataFrame(scaler.fit_transform(df_train),columns=df_train.columns)
df_test_scaled = pd.DataFrame(scaler.fit_transform(df_test),columns=df_test.columns)
```

In [46]:
```
df_train_scaled
```

Out[46]:

	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP	Legislative District	Expected Price
0	-0.064872	-0.751775	-0.872444	-0.588151	-0.728978	1.368115	-0.202356	-0.409185	0.965315
1	0.667998	-0.552517	-0.872444	1.700244	0.465955	-0.786575	-0.202356	-1.364864	-0.218853
2	1.400868	-0.751775	-0.057232	-0.588151	1.660888	-1.027054	-0.202356	1.160858	1.332815
3	1.400868	-0.751775	-0.057232	-0.588151	1.660888	-1.027054	-0.202356	-1.706177	1.332815
4	-0.064872	-0.353259	-0.872444	1.700244	-0.728978	-0.574954	-0.202356	0.614756	-0.860631
...	...	...	...	...	...	...	...	...	...
51459	-0.797743	-0.154001	-0.872444	-0.588151	-0.728978	-0.219045	-0.202356	1.092595	-0.749687
51460	0.301563	-0.154001	-0.872444	-0.588151	-0.728978	0.415819	-0.202356	-1.910966	-0.423020
51461	-0.797743	0.244515	-0.872444	1.700244	-0.728978	-0.334475	-0.202356	0.614756	-1.076354
51462	0.301563	-0.751775	0.757981	-0.588151	-0.728978	1.089159	-0.202356	1.024333	0.475314
51463	-0.064872	-0.751775	-0.872444	-0.588151	-0.728978	1.368115	-0.202356	0.751282	0.965315

51464 rows × 9 columns

In [47]:
```
df_test_scaled
```

Out[47]:

	Model Year	Make	Model	Electric Vehicle Type	Clean Alternative Fuel Vehicle (CAFV) Eligibility	Electric Range	Base MSRP	Legislative District	Expected Price

0	1.392467	-1.139327	-0.835942	-0.580193	-1.489071	-1.029595	-0.208307	0.072746	-0.479540
1	0.287682	-0.927823	-0.835942	-0.580193	0.148551	0.164835	-0.208307	0.754475	-0.919941
2	-0.448841	-0.716319	-0.835942	-0.580193	0.148551	-0.198272	-0.208307	-1.290711	-0.382106
3	-1.921887	-0.504815	-0.835942	-0.580193	0.148551	0.957937	5.416790	0.481783	-0.464340
4	0.655943	-0.504815	0.061080	-0.580193	0.148551	2.047257	-0.208307	0.890820	0.163524
...	...	...	...	...	...	...	...	...	...
12860	0.655943	-0.504815	1.855124	-0.580193	0.148551	1.731928	-0.208307	1.299858	2.190151
12861	-0.448841	-1.139327	-0.835942	-0.580193	0.148551	1.244600	-0.208307	1.231685	-1.005683
12862	-0.448841	-0.504815	-0.835942	-0.580193	0.148551	0.977048	-0.208307	-0.881674	0.553260
12863	-1.921887	0.129697	-0.835942	-0.580193	0.148551	-0.312937	-0.208307	1.095339	-1.083630
12864	-0.080580	-0.504815	0.061080	-0.580193	0.148551	1.024825	-0.208307	-0.881674	0.904023

12865 rows × 9 columns

In [48]:
```
X_train = df_train[[column for column in list(df_train.columns) if column!='Expected Price']]
X_test = df_test[[column for column in list(df_test.columns) if column!='Expected Price']]
y_train = df_train['Expected Price']
y_test = df_test['Expected Price']
X_train_scaled = df_train_scaled[[column for column in list(df_train_scaled.columns) if column!='Expected Price']]
X_test_scaled = df_test_scaled[[column for column in list(df_test_scaled.columns) if column!='Expected Price']]
y_train_scaled = df_train_scaled['Expected Price']
y_test_scaled = df_test_scaled['Expected Price']
```

In [49]:
```
LM = LinearRegression()
DTR = DecisionTreeRegressor()
RFR = RandomForestRegressor()
XGBR = XGBRegressor()
SVRR = SVR()
ETR = ExtraTreesRegressor()
ABR = AdaBoostRegressor()
GBR = GradientBoostingRegressor()
regressors = [LM,DTR,RFR,XGBR,SVRR,ETR,ABR,GBR]
Reg = ['LM','DTR','RFR','XGBR','SVRR','ETR','ABR','GBR']
R = [['Method','r2_score','RMSE','K-fold(score)']]
R2 = [['Method','r2_score','RMSE','K-fold(score)']]
for regressor in regressors:
    regressor.fit(X_train_scaled, y_train_scaled)
    reg = regressor.predict(X_train_scaled)
    reg2 = regressor.predict(X_test_scaled)
    cv = KFold(n_splits=10, random_state=1, shuffle=True)
    scores_tr = cross_val_score(regressor, X_train_scaled, y_train_scaled, scoring='neg_mean_squared_error',cv=cv, n_jobs=-1)
```

```
    scores_te = cross_val_score(regressor, X_test_scaled, y_test_scaled, scoring='neg_mean_squared_error',cv=cv, n_jobs=-1)
    R.append([Reg[regressors.index(regressor)],abs(r2_score(y_train_scaled, reg)),round(sqrt(mean_squared_error(y_train_scaled,
    R2.append([Reg[regressors.index(regressor)],abs(r2_score(y_test_scaled, reg2)),round(sqrt(mean_squared_error(y_test_scaled,
print('Test Data Metrics')
print(tabulate(R2,headers="firstrow"))
```

```
Test Data Metrics
Method      r2_score    RMSE    K-fold(score)
--------  ----------  ------  ---------------
LM          0.072021    0.48             0.75
DTR         0.821736    0.67             0.38
RFR         0.675491    0.65             0.37
XGBR        0.441505    0.6              0.37
SVRR        0.785487    0.67             0.56
ETR         0.203671    0.45             0.37
ABR         0.244271    0.56             0.63
GBR         0.294826    0.57             0.46
```