assignment2

August 23, 2023

[4]:
```
import pandas as pd
import numpy as np
temp = pd.read_csv("D:/ElectricCarData_Clean.csv")
print(temp.index)
```

```
RangeIndex(start=0, stop=103, step=1)
```

[9]:
```
print(temp.columns) # tells about number of columns
print(temp.shape) # rows,columns
```

```
Index(['Brand', 'Model', 'AccelSec', 'TopSpeed_KmH', 'Range_Km',
       'Efficiency_WhKm', 'FastCharge_KmH', 'RapidCharge', 'PowerTrain',
       'PlugType', 'BodyStyle', 'Segment', 'Seats', 'PriceEuro'],
      dtype='object')
(103, 14)
```

[10]:
```
temp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103 entries, 0 to 102
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Brand            103 non-null    object 
 1   Model            103 non-null    object 
 2   AccelSec         103 non-null    float64
 3   TopSpeed_KmH     103 non-null    int64  
 4   Range_Km         103 non-null    int64  
 5   Efficiency_WhKm  103 non-null    int64  
 6   FastCharge_KmH   103 non-null    object 
 7   RapidCharge      103 non-null    object 
 8   PowerTrain       103 non-null    object 
 9   PlugType         103 non-null    object 
 10  BodyStyle        103 non-null    object 
 11  Segment          103 non-null    object 
 12  Seats            103 non-null    int64  
 13  PriceEuro        103 non-null    int64  
```

```
dtypes: float64(1), int64(5), object(8)
memory usage: 11.4+ KB
```

[11]:
```
print(temp.head()) # prints the first 5 values
```

```
         Brand                          Model  AccelSec  TopSpeed_KmH  \
0        Tesla   Model 3 Long Range Dual Motor       4.6           233
1   Volkswagen                     ID.3 Pure      10.0           160
2     Polestar                             2       4.7           210
3          BMW                          iX3       6.8           180
4        Honda                            e       9.5           145

   Range_Km  Efficiency_WhKm FastCharge_KmH RapidCharge PowerTrain  \
0       450              161            940         Yes        AWD
1       270              167            250         Yes        RWD
2       400              181            620         Yes        AWD
3       360              206            560         Yes        RWD
4       170              168            190         Yes        RWD

     PlugType  BodyStyle Segment  Seats  PriceEuro
0  Type 2 CCS      Sedan       D      5      55480
1  Type 2 CCS  Hatchback       C      5      30000
2  Type 2 CCS   Liftback       D      5      56440
3  Type 2 CCS        SUV       D      5      68040
4  Type 2 CCS  Hatchback       B      4      32997
```

[12]:
```
print(temp.tail()) # prints the last 5 values
```

```
       Brand                          Model  AccelSec  TopSpeed_KmH  \
98    Nissan                    Ariya 63kWh       7.5           160
99     Audi     e-tron S Sportback 55 quattro       4.5           210
100   Nissan              Ariya e-4ORCE 63kWh       5.9           200
101   Nissan   Ariya e-4ORCE 87kWh Performance       5.1           200
102    Byton               M-Byte 95 kWh 2WD       7.5           190

     Range_Km  Efficiency_WhKm FastCharge_KmH RapidCharge PowerTrain  \
98        330              191            440         Yes        FWD
99        335              258            540         Yes        AWD
100       325              194            440         Yes        AWD
101       375              232            450         Yes        AWD
102       400              238            480         Yes        AWD

       PlugType  BodyStyle Segment  Seats  PriceEuro
98    Type 2 CCS  Hatchback       C      5      45000
99    Type 2 CCS        SUV       E      5      96050
100   Type 2 CCS  Hatchback       C      5      50000
101   Type 2 CCS  Hatchback       C      5      65000
102   Type 2 CCS        SUV       E      5      62000
```

[13]:
```
print(temp.describe())
```

```
         AccelSec  TopSpeed_KmH    Range_Km  Efficiency_WhKm       Seats  \
count  103.000000    103.000000  103.000000       103.000000  103.000000
mean     7.396117    179.194175  338.786408       189.165049    4.883495
std      3.017430     43.573030  126.014444        29.566839    0.795834
min      2.100000    123.000000   95.000000       104.000000    2.000000
25%      5.100000    150.000000  250.000000       168.000000    5.000000
50%      7.300000    160.000000  340.000000       180.000000    5.000000
75%      9.000000    200.000000  400.000000       203.000000    5.000000
max     22.400000    410.000000  970.000000       273.000000    7.000000

           PriceEuro
count     103.000000
mean    55811.563107
std     34134.665280
min     20129.000000
25%     34429.500000
50%     45000.000000
75%     65000.000000
max    215000.000000
```

[16]:
```
print(temp.mean())
```

```
AccelSec               7.396117
TopSpeed_KmH         179.194175
Range_Km             338.786408
Efficiency_WhKm      189.165049
Seats                  4.883495
PriceEuro          55811.563107
dtype: float64
```

```
C:\Users\Admin\AppData\Local\Temp/ipykernel_9916/1027130594.py:1: FutureWarning:
Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only valid
columns before calling the reduction.
  print(temp.mean())
```

[17]:
```
print(temp.median())
```

```
AccelSec               7.3
TopSpeed_KmH         160.0
Range_Km             340.0
Efficiency_WhKm      180.0
Seats                  5.0
PriceEuro          45000.0
dtype: float64
```

```
C:\Users\Admin\AppData\Local\Temp/ipykernel_9916/630863268.py:1: FutureWarning:
```

```
Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None')
is deprecated; in a future version this will raise TypeError.  Select only valid
columns before calling the reduction.
  print(temp.median())
```

[18]:
```
print(temp.mode())
```

```
    Brand          Model  AccelSec  TopSpeed_KmH  Range_Km  Efficiency_WhKm  \
0  Tesla   e-Soul 64 kWh       9.0           150       250            168.0
1     NaN            NaN       NaN           160       400              NaN

  FastCharge_KmH RapidCharge PowerTrain    PlugType BodyStyle Segment  Seats  \
0            230         Yes        AWD  Type 2 CCS       SUV       C    5.0
1            NaN         NaN        NaN         NaN       NaN     NaN    NaN

   PriceEuro
0    45000.0
1        NaN
```

[22]:
```
print(temp['AccelSec'].mean())
print(temp['TopSpeed_KmH'].mean())
print(temp['Range_Km'].mean())
print(temp['Efficiency_WhKm'].mean())
print(temp['Seats'].mean())
print(temp['PriceEuro'].mean())
```

```
7.396116504854368
179.19417475728156
338.7864077669903
189.16504854368932
4.883495145631068
55811.563106796115
```

[23]:
```
print(temp['AccelSec'].median())
print(temp['TopSpeed_KmH'].median())
print(temp['Range_Km'].median())
print(temp['Efficiency_WhKm'].median())
print(temp['Seats'].median())
print(temp['PriceEuro'].median())
```

```
7.3
160.0
340.0
180.0
5.0
45000.0
```

[24]:
```
print(temp['AccelSec'].mode())
print(temp['TopSpeed_KmH'].mode())
print(temp['Range_Km'].mode())
print(temp['Efficiency_WhKm'].mode())
print(temp['Seats'].mode())
print(temp['PriceEuro'].mode())
```

```
0    9.0
dtype: float64
0    150
1    160
dtype: int64
0    250
1    400
dtype: int64
0    168
dtype: int64
0    5
dtype: int64
0    45000
dtype: int64
```

[25]:
```
print(temp['AccelSec'].var())
print(temp['TopSpeed_KmH'].var())
print(temp['Range_Km'].var())
print(temp['Efficiency_WhKm'].var())
print(temp['Seats'].var())
print(temp['PriceEuro'].var())
```

```
9.104886731391584
1898.608985341709
15879.640205596796
874.1979821054638
0.6333523700742436
1165175373.7974494
```

[26]:
```
print(temp['AccelSec'].std())
print(temp['TopSpeed_KmH'].std())
print(temp['Range_Km'].std())
print(temp['Efficiency_WhKm'].std())
print(temp['Seats'].std())
print(temp['PriceEuro'].std())
```

```
3.0174304849311087
43.57303048149978
126.01444443236178
29.56683923089284
0.7958343860843433
```

```
34134.6652802902
```

[27]:
```
print(temp['AccelSec'].mad())
print(temp['TopSpeed_KmH'].mad())
print(temp['Range_Km'].mad())
print(temp['Efficiency_WhKm'].mad())
print(temp['Seats'].mad())
print(temp['PriceEuro'].mad())
```

```
2.2309548496559524
32.525591478932995
90.47224055047606
22.913940993496084
0.4722405504760114
23784.829672919215
```

[28]:
```
range1 = max(temp['AccelSec']) - min(temp['AccelSec'])
print(range1)
```

```
20.299999999999997
```

[29]:
```
Q1 = np.percentile(temp['AccelSec'],25)
Q3 = np.percentile(temp['AccelSec'],75)
print(Q1,Q3)
IQR = Q3-Q1
print(IQR)
```

```
5.1 9.0
3.9000000000000004
```

[30]:
```
unique,counts = np.unique(temp['AccelSec'],return_counts=True)
print(unique,counts)
```

```
[ 2.1  2.5  2.8  3.   3.2  3.4  3.5  3.7  3.8  4.   4.5  4.6  4.7  4.8
  4.9  5.   5.1  5.5  5.6  5.7  5.9  6.   6.2  6.3  6.5  6.6  6.8  6.9
  7.   7.3  7.5  7.6  7.8  7.9  8.1  8.2  8.3  8.5  8.7  8.8  9.   9.5
  9.6  9.7  9.8  9.9 10.  11.4 11.6 11.9 12.3 12.6 12.7 14.  22.4] [1 1 3 1 1 1
2 1 1 3 2 2 1 1 1 2 3 1 1 3 1 2 1 2 1 1 3 1 3 6 5 1 1 5 2 1 1
 2 1 1 7 2 1 2 1 2 4 2 1 2 2 1 1 1 1]
```

[ ]: