finalproject-1

August 1, 2025

[3]:
```
import pandas as pd
```

[5]:
```
df = pd.read_excel("FEV-data-Excel.xlsx")
```

[4]:
```
df.head()
```

[4]:
```
                      Car full name  Make                     Model  \
0            Audi e-tron 55 quattro  Audi            e-tron 55 quattro
1            Audi e-tron 50 quattro  Audi            e-tron 50 quattro
2             Audi e-tron S quattro  Audi             e-tron S quattro
3  Audi e-tron Sportback 50 quattro  Audi  e-tron Sportback 50 quattro
4  Audi e-tron Sportback 55 quattro  Audi  e-tron Sportback 55 quattro

   Minimal price (gross) [PLN]  Engine power [KM]  Maximum torque [Nm]  \
0                       345700                360                  664
1                       308400                313                  540
2                       414900                503                  973
3                       319700                313                  540
4                       357000                360                  664

        Type of brakes Drive type  Battery capacity [kWh]  Range (WLTP) [km]  \
0  disc (front + rear)        4WD                    95.0                438
1  disc (front + rear)        4WD                    71.0                340
2  disc (front + rear)        4WD                    95.0                364
3  disc (front + rear)        4WD                    71.0                346
4  disc (front + rear)        4WD                    95.0                447

   …  Permissable gross weight [kg]  Maximum load capacity [kg]  \
0  …                         3130.0                       640.0
1  …                         3040.0                       670.0
2  …                         3130.0                       565.0
3  …                         3040.0                       640.0
4  …                         3130.0                       670.0

   Number of seats  Number of doors  Tire size [in]  Maximum speed [kph]  \
0                5                5              19                  200
1                5                5              19                  190
```

```
2              5             5           20              210
3              5             5           19              190
4              5             5           19              200

   Boot capacity (VDA) [l]  Acceleration 0-100 kph [s]  \
0                    660.0                         5.7
1                    660.0                         6.8
2                    660.0                         4.5
3                    615.0                         6.8
4                    615.0                         5.7

   Maximum DC charging power [kW]  mean - Energy consumption [kWh/100 km]
0                             150                                  24.45
1                             150                                  23.80
2                             150                                  27.55
3                             150                                  23.30
4                             150                                  23.85

[5 rows x 25 columns]
```

[32]:

```python
# Task 1: A customer has a budget of 350,000 PLN and wants an EV with a minimum␣
↪range of 400 km.
# 1.a) Your task is to filter out EVs that meet these criteria.
filter_EV = df[((df['Minimal price (gross) [PLN]'] <= 350000) & (df['Range␣
↪(WLTP) [km]'] >= 400))]
filter_EV.head()
```

[32]:

```
                  Car full name     Make                Model  \
0         Audi e-tron 55 quattro     Audi    e-tron 55 quattro
8                       BMW iX3      BMW                  iX3
15  Hyundai Kona electric 64kWh  Hyundai  Kona electric 64kWh
18             Kia e-Niro 64kWh      Kia         e-Niro 64kWh
20             Kia e-Soul 64kWh      Kia         e-Soul 64kWh

    Minimal price (gross) [PLN]  Engine power [KM]  Maximum torque [Nm]  \
0                        345700                360                  664
8                        282900                286                  400
15                       178400                204                  395
18                       167990                204                  395
20                       160990                204                  395

         Type of brakes   Drive type  Battery capacity [kWh]  \
0   disc (front + rear)          4WD                    95.0
8   disc (front + rear)   2WD (rear)                    80.0
15  disc (front + rear)  2WD (front)                    64.0
18  disc (front + rear)  2WD (front)                    64.0
20  disc (front + rear)  2WD (front)                    64.0
```

```
    Range (WLTP) [km]  …  Permissable gross weight [kg]  \
0                 438  …                         3130.0
8                 460  …                         2725.0
15                449  …                         2170.0
18                455  …                         2230.0
20                452  …                         1682.0

    Maximum load capacity [kg]  Number of seats  Number of doors  \
0                        640.0                5                5
8                        540.0                5                5
15                       485.0                5                5
18                       493.0                5                5
20                       498.0                5                5

    Tire size [in]  Maximum speed [kph]  Boot capacity (VDA) [l]  \
0               19                  200                    660.0
8               19                  180                    510.0
15              17                  167                    332.0
18              17                  167                    451.0
20              17                  167                    315.0

    Acceleration 0-100 kph [s]  Maximum DC charging power [kW]  \
0                          5.7                              150
8                          6.8                              150
15                         7.6                              100
18                         7.8                              100
20                         7.9                              100

    mean - Energy consumption [kWh/100 km]
0                                    24.45
8                                    18.80
15                                   15.40
18                                   15.90
20                                   15.70

[5 rows x 25 columns]
```

[35]:

```python
# 1.b) Group them by the manufacturer (Make)
grouped_filter_EV = filter_EV.groupby('Make')
grouped_filter_EV.size().reset_index(name='EV count')
```

[35]:

	Make	EV count
0	Audi	1
1	BMW	1
2	Hyundai	1
3	Kia	2

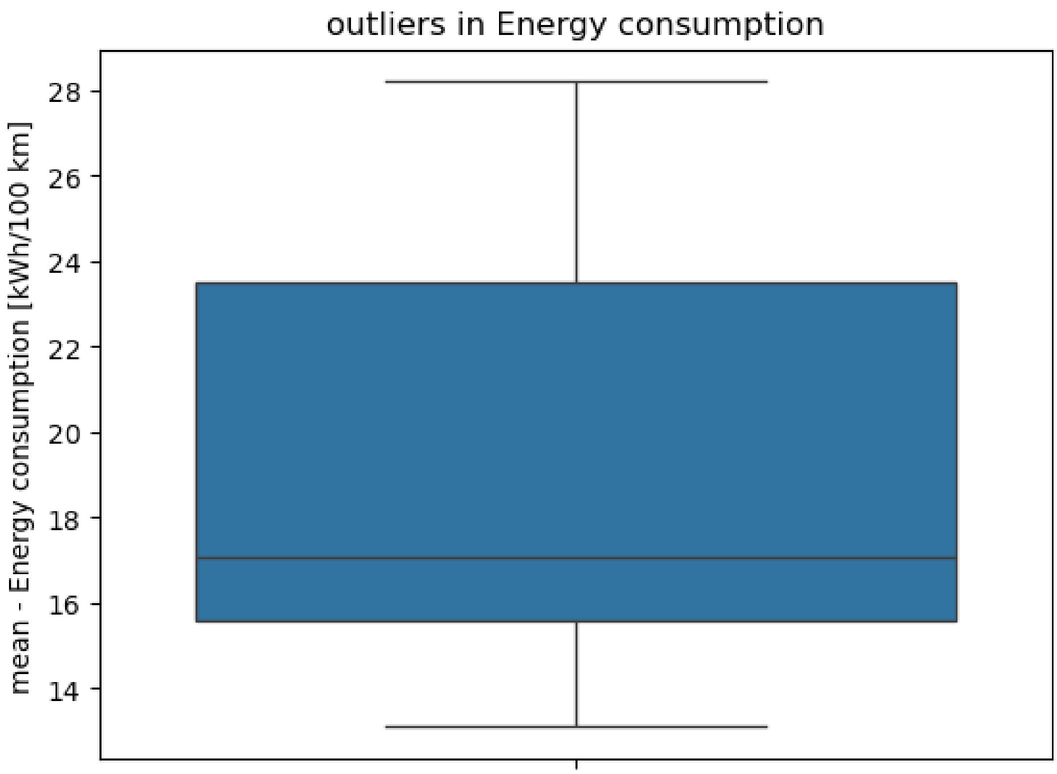
```
4  Mercedes-Benz         1
5          Tesla         3
6     Volkswagen         3
```

[37]:
```
# 1.c) Calculate the average battery capacity for each manufacturer.
grouped_mean = grouped_filter_EV['Battery capacity [kWh]'].mean().reset_index()
print(grouped_mean)
```

```
            Make  Battery capacity [kWh]
0           Audi               95.000000
1            BMW               80.000000
2        Hyundai               64.000000
3            Kia               64.000000
4  Mercedes-Benz               80.000000
5          Tesla               68.000000
6     Volkswagen               70.666667
```

[40]:
```
# Task 2: You suspect some EVs have unusually high or low energy consumption.
# Find the outliers in the mean - Energy consumption [kWh/100 km] column.
#Solution
# we can find outliers in python through many ways
#these are few methods When to Use Which?
#| Method  | Best For                     |
#| ------- | ---------------------------- |
#| IQR     | Most datasets, skewed values |
#| Z-Score | Normally distributed data    |
#| Boxplot | Quick visual detection       |

import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(y=df['mean - Energy consumption [kWh/100 km]'])
plt.title('outliers in Energy consumption')
plt.show()
```



[50]:

```
# using IQR for outliers
col = df['mean - Energy consumption [kWh/100 km]']

Q1 = col.quantile(0.25)
Q3 = col.quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 0.5 * IQR ## i have applied stricter threshold of 0.5 *␣
↪IQR(insted of 1.5 )
upper_bound = Q3 + 0.5 * IQR

outliers_iqr = df[(col < lower_bound) | (col > upper_bound)]
print(outliers_iqr[['Car full name', 'mean - Energy consumption [kWh/100 km]']])
```

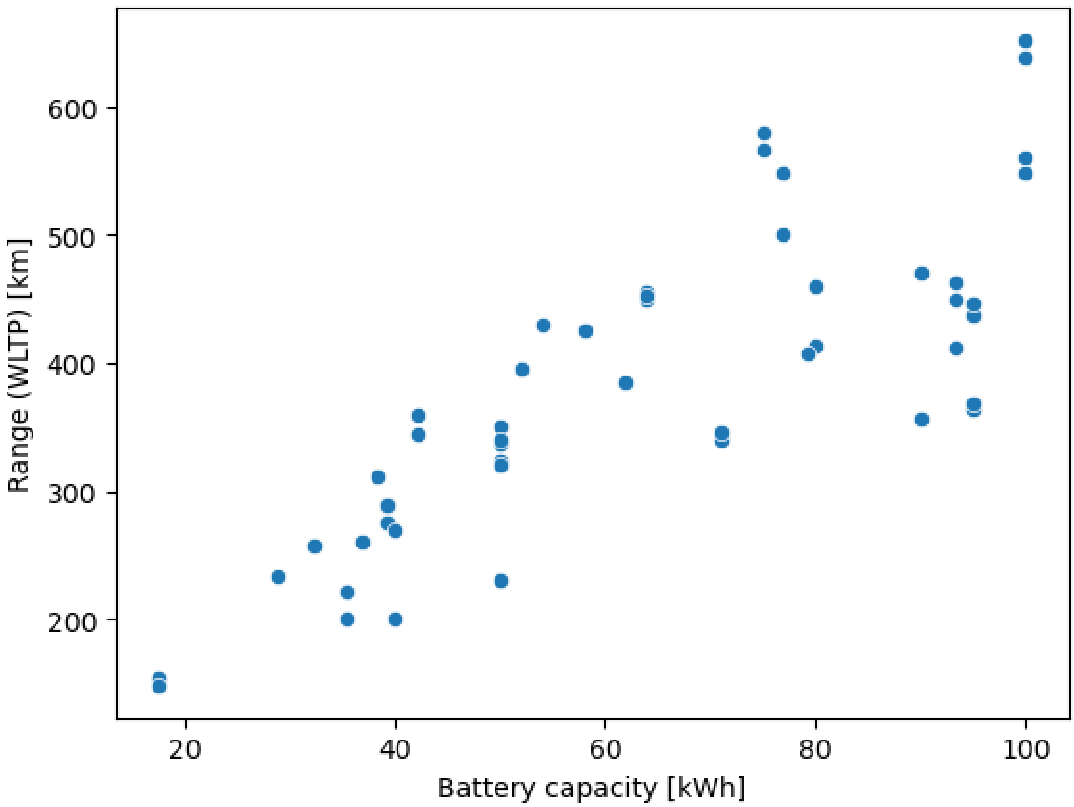
```
                 Car full name  mean - Energy consumption [kWh/100 km]
2        Audi e-tron S quattro                                   27.55
51  Mercedes-Benz EQV (long)                                     28.20
```

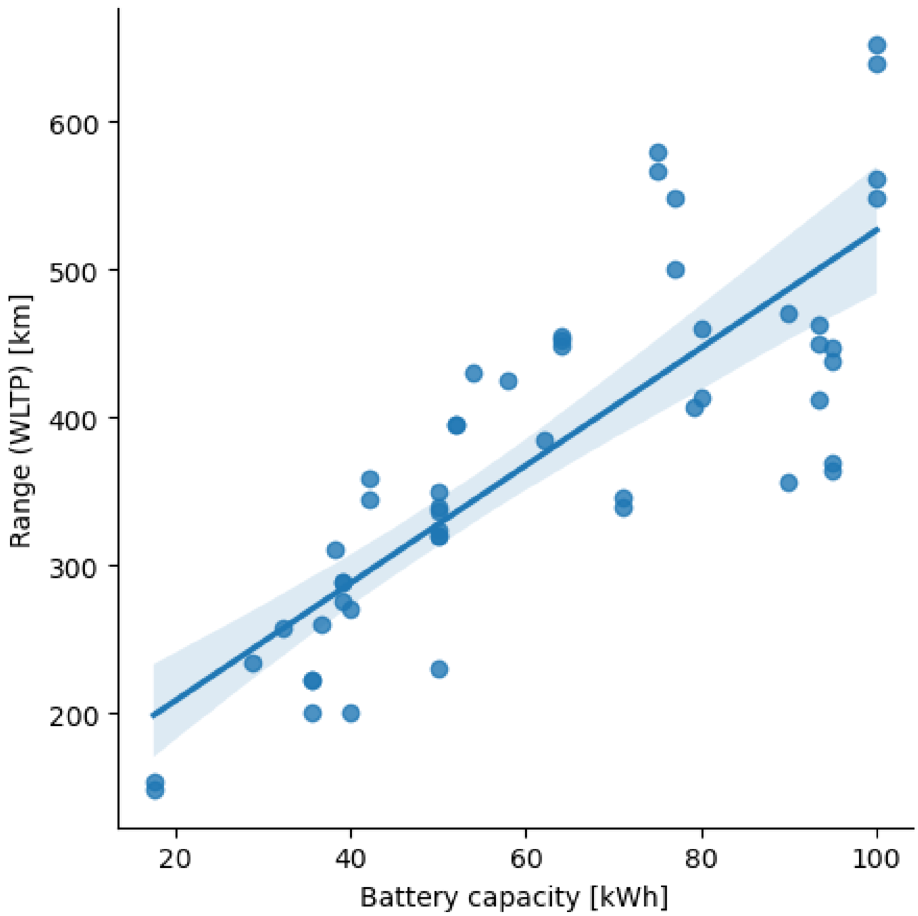
[55]:

```
#Task 3: Your manager wants to know if there's a strong relationship between␣
↪battery capacity and range.
# a) Create a suitable plot to visualize.
```

```python
# Solution
#For Two Numeric Columns :- Use a Scatter Plot(Shows correlation, trends, and␣
↪clusters.)
#For Categorical vs Numeric:- Use Box Plot or Violin Plot(Compares␣
↪distributions across categories , Use violin plot for more detail:)
#For Two Categorical Columns:- Use Heatmap or Count Plot
#Bonus: Correlation Matrix:- To find overall correlation between all numeric␣
↪columns:
import seaborn as sns
sns.scatterplot(x=df['Battery capacity [kWh]'],y= df['Range (WLTP) [km]'])
sns.lmplot(x='Battery capacity [kWh]', y='Range (WLTP) [km]', data=df) #␣
↪lmplot= linear model plot for regression line
```

[55]: <seaborn.axisgrid.FacetGrid at 0x14cf50137d0>





[ ]:

```
# b) Highlight any insights.
'''
Insights from the Scatter Plot: Battery Capacity vs Range
1)Positive Correlation:
*There's a clear positive relationship between battery capacity and range - as␣
↪battery size increases, the EV's range also tends to increase.
*The regression line confirms this trend.

2)Diminishing Returns:
*At higher battery capacities (e.g., >80 kWh), the increase in range becomes␣
↪less steep, indicating diminishing efficiency gains.

3)Outliers:

*A few vehicles with relatively high battery capacities but lower range may␣
↪indicate inefficient models or heavier cars.
*Similarly, some cars with lower battery size and high range suggest␣
↪efficiency-optimized EVs.
```

```
'''
```

[6]:
```
'''
Task 4: Build an EV recommendation class.
The class should allow users to input their budget, desired range, and battery␣
↪capacity.
The class should then return the top three EVs matching their criteria.
'''


class EVRecommender:
    def __init__(self, dataframe):
        self.df = dataframe

    def recommend(self, budget, min_range, min_battery):
        # Filter based on user input
        filtered = self.df[
            (self.df['Minimal price (gross) [PLN]'] <= budget) &
            (self.df['Range (WLTP) [km]'] >= min_range) &
            (self.df['Battery capacity [kWh]'] >= min_battery)
        ]
        # Sort and return top 3
        top_evs = filtered.sort_values(by='Range (WLTP) [km]', ascending=False).
↪head(3)
        return top_evs[['Make', 'Model', 'Minimal price (gross) [PLN]', 'Range␣
↪(WLTP) [km]', 'Battery capacity [kWh]']]

# Instantiate and test the recommender
recommender = EVRecommender(df)
recommended_evs = recommender.recommend(budget=350000, min_range=400,␣
↪min_battery=60)
recommended_evs
```

[6]:
```
          Make                 Model  Minimal price (gross) [PLN]  \
40       Tesla    Model 3 Long Range                       235490
41       Tesla  Model 3 Performance                       260490
48  Volkswagen            ID.3 Pro S                       179990

    Range (WLTP) [km]  Battery capacity [kWh]
40                580                    75.0
41                567                    75.0
48                549                    77.0
```

[9]:
```
# Task 5: Inferential Statistics - Hypothesis Testing: Test whether there is a␣
↪significant
# difference in the average Engine power [KM] of vehicles manufactured by two␣
↪leading
```

```
# manufacturers i.e. Tesla and Audi. What insights can you draw from the test␣
↪results?
# Recommendations and Conclusion: Provide actionable insights based on your␣
↪analysis.
# (Conduct a two sample t-test using ttest_ind from scipy.stats module)

from scipy.stats import ttest_ind
# Filter for Tesla and Audi vehicles only
tesla_power = df[df['Make'] == "Tesla"]['Engine power [KM]'].dropna()
audi_power = df[df['Make'] == "Audi"]['Engine power [KM]'].dropna()

# Two-sample t-test (Welch's t-test by default)
t_stat, p_value = ttest_ind(tesla_power, audi_power, equal_var=False)

print(f"T-statistic: {t_stat:.2f}")
print(f"P-value: {p_value:.3f}")

if p_value < 0.05:
    print("There is a statistically significant difference in average engine␣
↪power between Tesla and Audi.")
else:
    print("No statistically significant difference in average engine power␣
↪between Tesla and Audi.")
```

```
T-statistic: 1.79
P-value: 0.107
No statistically significant difference in average engine power between Tesla
and Audi.
```

[ ]:
```
# Task 6:  Video explain
# https://drive.google.com/file/d/1trY5DrK_jIYQh_DmA6Gw9OEof1-Dr8nD/view?
↪usp=drive_link
```