ELECTRIC VEHICLE DATA ANALYSIS PROJECT

COURSE-5 PYTHON

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
FEV_df=pd.read_excel("FEV-data-Excel.xlsx")
```

Data Preparation

```
Up_FEV_df=FEV_df.dropna(how="all")  #drops rows only if all values are NaN
```

```
Up_FEV_df.fillna({"Type of brakes":'NA',})
Up_FEV_df.head()
```

	Car full name	Make	Model	Minimal price (gross) [PLN]	Engine power [KM]	Maximum torque [Nm]	Type of brakes	Drive type	Battery capacity [kWh]	Range (WLTP) [km]	...	Permissable gross weight [kg]	Maximum load capacity [kg]	Number of seats	Number of doors	Tire size [in]	Maximum speed [kph]	Boot capacity (VDA) [l]	Accele 0-1
0	Audi e-tron 55 quattro	Audi	e-tron 55 quattro	345700	360	664	disc (front + rear)	4WD	95.0	438	...	3130.0	640.0	5	5	19	200	660.0	
1	Audi e-tron 50 quattro	Audi	e-tron 50 quattro	308400	313	540	disc (front + rear)	4WD	71.0	340	...	3040.0	670.0	5	5	19	190	660.0	
2	Audi e-tron S quattro	Audi	e-tron S quattro	414900	503	973	disc (front + rear)	4WD	95.0	364	...	3130.0	565.0	5	5	20	210	660.0	
3	Audi e-tron Sportback 50 quattro	Audi	e-tron Sportback 50 quattro	319700	313	540	disc (front + rear)	4WD	71.0	346	...	3040.0	640.0	5	5	19	190	615.0	
4	Audi e-tron Sportback 55 quattro	Audi	e-tron Sportback 55 quattro	357000	360	664	disc (front + rear)	4WD	95.0	447	...	3130.0	670.0	5	5	19	200	615.0	

```
# Column names changed to make them shorter and easy to use
Up_FEV_df.columns=["Car Full Name", "Make","Model","Minimal_Price","Engine_Power_km","Maximum_torque_Nm","Type_of_brakes","Drive_type","Bat_capacity_kwh",
                   "Range_km","WHeelbase_cm","Length_cm","Width_cm","Height_cm","Empty_weight_kg","P_gross_weight_kg","Max_load_capa_kg","Num_of_seats",
                   "Num_of_doors","Tire_size_in","Max_speed_kph","Boot_capacity_liter","Acceleration_kph","Charging_power_kw","Mean_EC"]
```

```
Up_FEV_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53 entries, 0 to 52
Data columns (total 25 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Car Full Name        53 non-null     object 
 1   Make                 53 non-null     object 
 2   Model                53 non-null     object 
 3   Minimal_Price        53 non-null     int64  
 4   Engine_Power_km      53 non-null     int64  
 5   Maximum_torque_Nm    53 non-null     int64  
 6   Type_of_brakes       52 non-null     object 
 7   Drive_type           53 non-null     object 
 8   Bat_capacity_kwh     53 non-null     float64
 9   Range_km             53 non-null     int64  
 10  WHeelbase_cm         53 non-null     float64
 11  Length_cm            53 non-null     float64
 12  Width_cm             53 non-null     float64
 13  Height_cm            53 non-null     float64
 14  Empty_weight_kg      53 non-null     int64  
 15  P_gross_weight_kg    45 non-null     float64
 16  Max_load_capa_kg     45 non-null     float64
 17  Num_of_seats         53 non-null     int64  
 18  Num_of_doors         53 non-null     int64  
 19  Tire_size_in         53 non-null     int64  
 20  Max_speed_kph        53 non-null     int64  
 21  Boot_capacity_liter  52 non-null     float64
 22  Acceleration_kph     50 non-null     float64
 23  Charging_power_kw    53 non-null     int64  
 24  Mean_EC              44 non-null     float64
dtypes: float64(10), int64(10), object(5)
```

```
memory usage: 10.5+ KB
```

```
### Filled up coloumns having NaN values with 0.0 or NA
Mod_FEV_df=Up_FEV_df.fillna({"Type_of_brakes":"NA","P_gross_weight_kg":0.0,"Max_load_capa_kg":0.0,"Boot_capacity_liter":0.0,"Acceleration_kph":0.0,"Mean_EC":0.0})
```

```
Mod_FEV_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53 entries, 0 to 52
Data columns (total 25 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Car Full Name        53 non-null     object 
 1   Make                 53 non-null     object 
 2   Model                53 non-null     object 
 3   Minimal_Price        53 non-null     int64  
 4   Engine_Power_km      53 non-null     int64  
 5   Maximum_torque_Nm    53 non-null     int64  
 6   Type_of_brakes       53 non-null     object 
 7   Drive_type           53 non-null     object 
 8   Bat_capacity_kwh     53 non-null     float64
 9   Range_km             53 non-null     int64  
 10  WHeelbase_cm         53 non-null     float64
 11  Length_cm            53 non-null     float64
 12  Width_cm             53 non-null     float64
 13  Height_cm            53 non-null     float64
 14  Empty_weight_kg      53 non-null     int64  
 15  P_gross_weight_kg    53 non-null     float64
 16  Max_load_capa_kg     53 non-null     float64
 17  Num_of_seats         53 non-null     int64  
 18  Num_of_doors         53 non-null     int64  
 19  Tire_size_in         53 non-null     int64  
 20  Max_speed_kph        53 non-null     int64  
 21  Boot_capacity_liter  53 non-null     float64
 22  Acceleration_kph     53 non-null     float64
 23  Charging_power_kw    53 non-null     int64  
 24  Mean_EC              53 non-null     float64
dtypes: float64(10), int64(10), object(5)
memory usage: 10.5+ KB
```

Task 1: A customer has a budget of 350,000 PLN and wants an EV with a minimum range of 400 km.

(a) Your task is to filter out EVs that meet these criteria.

```
filtered_EVs=Mod_FEV_df[(Mod_FEV_df['Minimal_Price']<=350000) & (Mod_FEV_df['Range_km']>=400)]  #This filters rows where Minimal_price is less than or equal to 350,(
filtered_EVs=filtered_EVs[['Car Full Name','Make','Model','Minimal_Price','Range_km','Bat_capacity_kwh' ]]          # and range is equal to 400 or above
filtered_EVs.sort_values("Minimal_Price",ascending=False)
```

	Car Full Name	Make	Model	Minimal_Price	Range_km	Bat_capacity_kwh
0	Audi e-tron 55 quattro	Audi	e-tron 55 quattro	345700	438	95.0
22	Mercedes-Benz EQC	Mercedes-Benz	EQC	334700	414	80.0
8	BMW iX3	BMW	iX3	282900	460	80.0
41	Tesla Model 3 Performance	Tesla	Model 3 Performance	260490	567	75.0
40	Tesla Model 3 Long Range	Tesla	Model 3 Long Range	235490	580	75.0
49	Volkswagen ID.4 1st	Volkswagen	ID.4 1st	202390	500	77.0
39	Tesla Model 3 Standard Range Plus	Tesla	Model 3 Standard Range Plus	195490	430	54.0
48	Volkswagen ID.3 Pro S	Volkswagen	ID.3 Pro S	179990	549	77.0
15	Hyundai Kona electric 64kWh	Hyundai	Kona electric 64kWh	178400	449	64.0
18	Kia e-Niro 64kWh	Kia	e-Niro 64kWh	167990	455	64.0
20	Kia e-Soul 64kWh	Kia	e-Soul 64kWh	160990	452	64.0
47	Volkswagen ID.3 Pro Performance	Volkswagen	ID.3 Pro Performance	155890	425	58.0

b) Group them by the manufacturer (Make).

```
Group_make=filtered_EVs.groupby("Make")["Range_km"].mean()  # This groups the manufacturer based on the aggregate range
Group_make.sort_values(ascending=False)
```

Make	Range_km
Tesla	525.666667
Volkswagen	491.333333
BMW	460.000000
Kia	453.500000
Hyundai	449.000000
Audi	438.000000
Mercedes-Benz	414.000000

dtype: float64

(c) Calculate the average battery capacity for each manufacturer.

```
Avg_bat_Capacity=filtered_EVs.groupby("Make")["Bat_capacity_kwh"].mean()
Avg_bat_Capacity.sort_values(ascending=False)
```

Make	Bat_capacity_kwh
Audi	95.000000
BMW	80.000000
Mercedes-Benz	80.000000
Volkswagen	70.666667
Tesla	68.000000
Hyundai	64.000000
Kia	64.000000

dtype: float64

INSIGHTS

1.Multiple VW models (ID.3, ID.4) offer 450-550 km range at under 200k PLN.

2.Volkswagen ID.3 Pro S achieves 549 km range on 77 kWh, showing high energy efficiency compared to larger battery vehicles.

3.Hyundai and Kia offer 450+ km range with 64 kWh batteries priced between 160k-180k PLN.

4.Tesla has an Average range of approx. 526 km, well above other brands.

5.With an average range of 491 km(approx), VW is emerging as a serious competitor to Tesla in the affordable EV space.

6.Audi (438 km) and Mercedes-Benz (414 km) have lower average ranges despite higher prices.

7.Lower-priced brands like Tesla (Model 3) and Volkswagen offer higher average range than premium brands like Audi and Mercedes.

8.Audi leads with the highest average battery capacity among selected EVs.

9.BMW and Mercedes-Benz offer a good balance between battery size and premium branding.

10.Volkswagen and Tesla provide relatively smaller battery capacities but compensate with longer ranges, especially Tesla.

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.(16 Marks)

```
Mod_FEV_df[['Car Full Name','Make','Model','Mean_EC' ]].head(10)
```

	Car Full Name	Make	Model	Mean_EC
0	Audi e-tron 55 quattro	Audi	e-tron 55 quattro	24.45
1	Audi e-tron 50 quattro	Audi	e-tron 50 quattro	23.80
2	Audi e-tron S quattro	Audi	e-tron S quattro	27.55
3	Audi e-tron Sportback 50 quattro	Audi	e-tron Sportback 50 quattro	23.30
4	Audi e-tron Sportback 55 quattro	Audi	e-tron Sportback 55 quattro	23.85
5	Audi e-tron Sportback S quattro	Audi	e-tron Sportback S quattro	27.20
6	BMW i3	BMW	i3	13.10
7	BMW i3s	BMW	i3s	14.30
8	BMW iX3	BMW	iX3	18.80
9	Citroën ë-C4	Citroën	ë-C4	0.00

```
out_FEV_df=Mod_FEV_df[Mod_FEV_df['Mean_EC']!=0]    #This will remove all rows having zero in the Mean_FC column. Otherwise these will turn out to be outliers and af
out_FEV_df.head(10)
```

	Car Full Name	Make	Model	Minimal_Price	Engine_Power_km	Maximum_torque_Nm	Type_of_brakes	Drive_type	Bat_capacity_kwh	Range_km	...	P_gross_weight_kg
0	Audi e-tron 55 quattro	Audi	e-tron 55 quattro	345700	360	664	disc (front + rear)	4WD	95.0	438	...	3130.0
1	Audi e-tron 50 quattro	Audi	e-tron 50 quattro	308400	313	540	disc (front + rear)	4WD	71.0	340	...	3040.0
2	Audi e-tron S quattro	Audi	e-tron S quattro	414900	503	973	disc (front + rear)	4WD	95.0	364	...	3130.0
3	Audi e-tron Sportback 50 quattro	Audi	e-tron Sportback 50 quattro	319700	313	540	disc (front + rear)	4WD	71.0	346	...	3040.0
4	Audi e-tron Sportback 55 quattro	Audi	e-tron Sportback 55 quattro	357000	360	664	disc (front + rear)	4WD	95.0	447	...	3130.0
5	Audi e-tron Sportback S quattro	Audi	e-tron Sportback S quattro	426200	503	973	disc (front + rear)	4WD	95.0	369	...	3130.0
6	BMW i3	BMW	i3	169700	170	250	disc (front + rear)	2WD (rear)	42.2	359	...	1730.0
7	BMW i3s	BMW	i3s	184200	184	270	disc (front + rear)	2WD (rear)	42.2	345	...	1730.0
8	BMW iX3	BMW	iX3	282900	286	400	disc (front + rear)	2WD (rear)	80.0	460	...	2725.0
10	DS DS3 Crossback e-tense	DS	DS3 Crossback e-tense	159900	136	260	disc (front + rear)	2WD (front)	50.0	320	...	1975.0

10 rows × 25 columns

```
out_FEV_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 44 entries, 0 to 52
Data columns (total 25 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Car Full Name       44 non-null     object 
 1   Make                44 non-null     object 
 2   Model               44 non-null     object 
 3   Minimal_Price       44 non-null     int64  
 4   Engine_Power_km     44 non-null     int64  
 5   Maximum_torque_Nm   44 non-null     int64  
 6   Type_of_brakes      44 non-null     object 
 7   Drive_type          44 non-null     object 
 8   Bat_capacity_kwh    44 non-null     float64
 9   Range_km            44 non-null     int64  
 10  WHeelbase_cm        44 non-null     float64
 11  Length_cm           44 non-null     float64
 12  Width_cm            44 non-null     float64
 13  Height_cm           44 non-null     float64
 14  Empty_weight_kg     44 non-null     int64  
 15  P_gross_weight_kg   44 non-null     float64
 16  Max_load_capa_kg    44 non-null     float64
 17  Num_of_seats        44 non-null     int64  
 18  Num_of_doors        44 non-null     int64  
 19  Tire_size_in        44 non-null     int64  
 20  Max_speed_kph       44 non-null     int64  
 21  Boot_capacity_liter 44 non-null     float64
 22  Acceleration_kph    44 non-null     float64
 23  Charging_power_kw   44 non-null     int64  
 24  Mean_EC             44 non-null     float64
dtypes: float64(10), int64(10), object(5)
memory usage: 8.9+ KB
```

```
out_FEV_df['Mean_EC'].mean()
```

```
np.float64(18.994318181818183)
```

```
out_FEV_df['Mean_EC'].median()
```

```
17.05
```

Since mean and median are very close to each other, but still there is some difference between mean and median, so there can be some outliers.

```
out_FEV_df['Mean_EC'].var()   #Variance of Mean_EC column
```

```
19.520955338266386
```

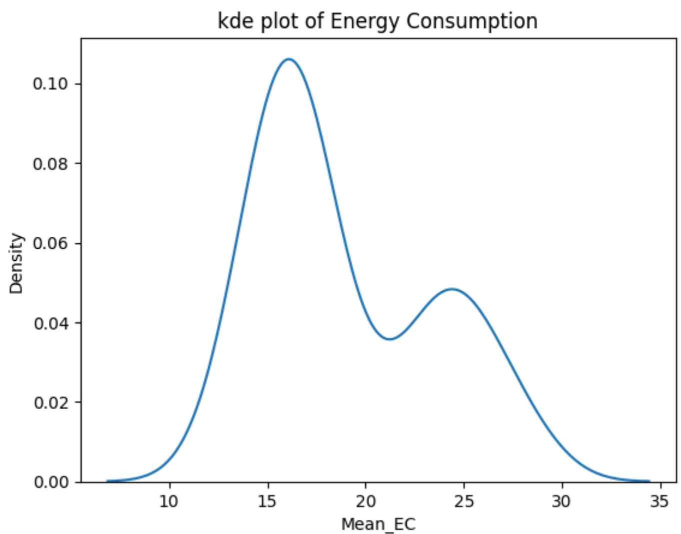
Double-click (or enter) to edit

```
out_FEV_df['Mean_EC'].std()   #Standard deviation of Mean_EC column
```

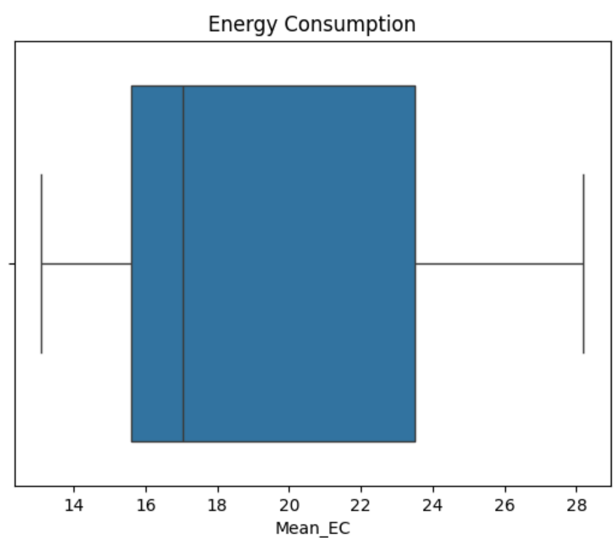
```
4.418252520880443
```

```
sns.kdeplot(x='Mean_EC',data=out_FEV_df)
plt.title("kde plot of Energy Consumption")
```

```
Text(0.5, 1.0, 'kde plot of Energy Consumption')
```




```
sns.boxplot(x=out_FEV_df['Mean_EC'])
plt.title("Energy Consumption")
plt.show()
```



To find outliers in the Mean_EC column. We need to find Lower Bound(LB), Upper Bound(UB),Interquartile range, quartile 1(q1) and quartile 2(q2). The formular for IQR

```
IQR=q3-q1
```

```
q1=out_FEV_df['Mean_EC'].quantile(0.25) #formula to find q1
q3=out_FEV_df['Mean_EC'].quantile(0.75) #formula to find q3
```

```
q1
```

```
np.float64(15.6)
```

```
q3
```

```
np.float64(23.5)
```

```
IQR=q3-q1 #calculation of IQR
IQR
```

```
np.float64(7.9)
```

```
LB=q1-1.5*IQR
UB=q3+1.5*IQR

LB,UB                          # So any value below LB and above UB will be an outlier
```

```
(np.float64(3.7499999999999982), np.float64(35.35))
```

```
out_FEV_df['Mean_EC'].min()
```

```
13.1
```

```
out_FEV_df['Mean_EC'].max()
```

```
28.2
```

```
# so we can see that there is no value below the lower bound value of 3.74 in the mean_EC column.
# The maximum value in the Mean_EC column fall below the upper bound. so there is no outlier on this side as well
```

```
Outliers=out_FEV_df[(out_FEV_df['Mean_EC']<LB) | (out_FEV_df['Mean_EC']>UB)]     # This will extract rows having values below LB and above UB if there is any.
Outliers=Outliers[["Car Full Name","Make","Model","Mean_EC"]]
Outliers
```

Car Full Name	Make	Model	Mean_EC

INSIGHTS

1.All EVs fall within expected range (13 to 28 kWh/100 km).

2.Skewed distibution, a few high-consumption EVs may be influencing the mean more than the median.

3.Highest mean energy consumption: 28.2 kWh/100 km.Suggests this EV might be heavy or performance-oriented.

4.Lowest consumption: 13.1 kWh/100 km,Indicates exceptional efficiency.Left boundary near 13.1 — these may be city-oriented EVs.

5.Energy consumption is centered around 17 kWh/100 km — most EVs are energy efficient.

6.No visual or statistical outliers.

Since no rows apears in the result, so it can be concluded that there is no outlier in the Mean_EC

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.

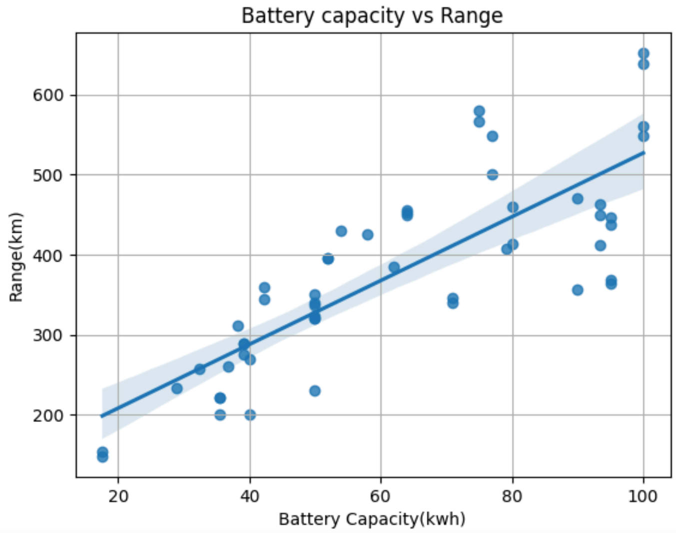
```
Mod_FEV_df['Bat_capacity_kwh'].corr(Mod_FEV_df['Range_km']) #the correlation value of 0.81 represents
                                                            #that there is a strong positive relationship between Battery capacity and Rnage
```

```
np.float64(0.8104385771936846)
```

```
correlation=np.corrcoef(Mod_FEV_df['Bat_capacity_kwh'],Mod_FEV_df['Range_km'])[0,1]
correlation
```

```
np.float64(0.8104385771936846)
```

```
sns.regplot(data=Mod_FEV_df,x='Bat_capacity_kwh',y='Range_km')  #Scatter plot has been used here to check for a linear or non-linear relationship
plt.title("Battery capacity vs Range")                          #between battery capacity and range
plt.xlabel("Battery Capacity(kwh)")
plt.ylabel("Range(km)")
plt.grid()
plt.show()
```



b) Highlight any insights.

Insights:

1.Correlation Coefficient of 0.81 Indicates a strong positive linear relationship between battery capacity and driving range.

2.(0.81) coeff means larger batteries provide longer driving range.

3.The plot shows a clear upward trend—as battery capacity increases, so does range.

5.Some spread at higher capacities suggests lower returns or it may be due to influence of other factors like weight, and aerodynamics.

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. (8+8 Marks)

```
class EVrecommender:    #defined a class

  def __init__(self,Mod_FEV_df): #init method initialize the object
    self.df=Mod_FEV_df          #assigns the dataset to the object

  def recommend(self,budget,desired_range,battery_capacity):  # Function named "recommend" created inside the class with three parameters
      recommended_EVs=self.df[(self.df['Minimal_Price']<=budget) & (self.df['Range_km']>=desired_range) & # filters dataset based on user inputs
       (self.df['Bat_capacity_kwh']>=battery_capacity)]

      Top3_EVs=recommended_EVs.sort_values(['Minimal_Price'],ascending=True) #Sorts the filtered dataset by price
      return Top3_EVs[['Car Full Name','Make','Model','Minimal_Price','Bat_capacity_kwh','Range_km']].head(3) #return top 3 EVs and store them in Top3_EVs
```

```
EVs=EVrecommender(Mod_FEV_df) #creates an object EVs inside the class and loads the dataset
Recommendations=EVs.recommend(budget=500000,desired_range=400,battery_capacity=80) #calls the recommend method with customer preferences
Recommendations
```

	Car Full Name	Make	Model	Minimal_Price	Bat_capacity_kwh	Range_km
8	BMW iX3	BMW	iX3	282900	80.0	460
22	Mercedes-Benz EQC	Mercedes-Benz	EQC	334700	80.0	414
0	Audi e-tron 55 quattro	Audi	e-tron 55 quattro	345700	95.0	438

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) (16 Marks)

Hypothesis Testing : Comparison of Engine Power of Tesla and Audi.

Objective: To test whether there is statistically significant difference in the average engine power of EVs manufactured by Tesla and Audi.

Null Hypothesis (Ho):

There is no significant difference in the average engine power of vehicles manufactured by Tesla and Audi.

Alternative Hypothesis(H1)

There is significant difference in the average engine power of vehicles manufactured by Tesla and Audi.

```
from scipy.stats import ttest_ind
```

```
Mod_FEV_df[Mod_FEV_df['Make'] == 'Audi'][['Make','Engine_Power_km']]
```

	Make	Engine_Power_km
0	Audi	360
1	Audi	313
2	Audi	503
3	Audi	313
4	Audi	360
5	Audi	503

```
Mod_FEV_df[Mod_FEV_df['Make']=='Tesla'][['Make','Engine_Power_km']]
```

	Make	Engine_Power_km
39	Tesla	285
40	Tesla	372
41	Tesla	480
42	Tesla	525
43	Tesla	772
44	Tesla	525
45	Tesla	772

```
# Codes to filter the data frame Mod_FEV_df to select the engine power
Tesla_Power=Mod_FEV_df[Mod_FEV_df['Make']=='Tesla']['Engine_Power_km'] # This code filters rwos where the brand is Tesla and creates a series of engine powers of Tes
Audi_Power=Mod_FEV_df[Mod_FEV_df['Make']=='Audi']['Engine_Power_km']# This code filters rows where the brand is Audi and creates a series of engine powers of Audi
```

```
Average_Tesla=Tesla_Power.mean()
Average_Audi=Audi_Power.mean()
print(Average_Tesla,Average_Audi)
```

```
533.0 392.0
```

```
#two sample independent t test using ttest_ind
t_stat, p_value=ttest_ind(Tesla_Power,Audi_Power,equal_var=False)  #ttest_ind compares the means of two independent groups i.e. Tesla_Power and Audi_Power
                                                                   #equal_var=false doesnot assume varinces of
```

```
print("T-Statistic:",t_stat)
print("P-value:",p_value)
```

```
T-Statistic: 1.7939951827297178
P-value: 0.10684105068839565
```

```
alpha=0.05
#i.e. 5% siginicance level
# p-value less than 0.05 suggests strong evidence against the null hypothesis
# Large p-value suggests weak evidence against the null hypothesis
if p_value<alpha:
    print("Null hypotheis is rejected:There is significant difference in engine power")
else:
  print("Fail to reject the null hypothesis:There is no significant difference in engine power.")
```

```
Fail to reject the null hypothesis:There is no significant difference in engine power.
```

INSIGHTS

1. Tesla has higher averge engine power , but the difference is not statistically significant.

RECOMMENDATIONS:

1. Promote Tesla models as they offer superior range at competitive prices.
2. Highlight value models like VW ID.3 Pro S and Kia -e niro provide excellent range under 200000 PLN. They are ideal for budget conscious customers.
3. Avoidn over priced low range models like Audi e-tron in a value market.
4. VW and Tesla outperform others in average range , so these can be marketed for long range EV seekers.
5. Models with energy usage greater than 28 KWH/100KM are inefficient, manufacturer should analyze these products or consider price change for these EVs.
6. Promote EVS having less than 17kwh/100km energy consumption for energy conscious customers.
7. Battery capacity can be used as key selling point.
8. Companies should focus on efficiency rather than simply increasing battery size.
9. Since the difference in engine power of Tesla and Audi is not statiscally significant so companies should emphasize other features like range, features or efficiency.

Project Video Explanation Link

https://www.google.com/url?
q=https%3A%2F%2Fdrive.google.com%2Ffile%2Fd%2F1SZuUFeRedIwGGYefmbj0KGHpGQs0X7cy%2Fview%3Fusp%3Dsharing