Waze Project - Inspect and analyze data

Your team is still in the early stages of their project to develop a machine learning model to predict user churn.

You have received notice that your project proposal has been approved and your team has been given access to Waze's user data. To get clear insights, the data must be inspected, organized, and prepared for analysis.

You discover two new emails in your inbox: one from May Santner, and one from your teammate, Chidi Ga. In the email, May asks for your help reviewing the data and completing a code notebook, and Chidi shares the details of the notebook. Review the emails, then follow the provided instructions to complete the PACE strategy document, the code notebook, and the executive summary.

Briefing: "Until we finish our previous project, *there is no need to do a full EDA on our new user data. We'll get to that soon. Meanwhile, do you mind reviewing the imported data for the team? It would be fantastic if you could include a summary of the data types for each variable, where missing values exist in the data, key descriptive statistics, and anything else code-related you think is worth sharing in the notebook. I haven't had a chance to explore the data, so I really appreciate you getting an early start on this"*

Task 1. Understand the situation

- *How can you best prepare to understand and organize the provided driver data?*

Step 1) Pre processing and cleaning

I load the dataset waze_dataset.csv into pycharm, creating a dataframe will help me to conduct data manipulation, exploratory data analysis (EDA), and statistical activities.

input:

```
import pandas as pd
import numpy as np
```

```python
df =
pd.read_csv("C:\\Users\\Lenovo\\Downloads\\waze_dataset.csv")

print(df.head(10))
```

output:

	ID	label	sessions	drives	total_sessions	n_days_after_onboarding	total_navigations_fav1	total_navigations_fav2	driven_km_drives	duration_minutes_drives
0	0	retained	283	226	296.748273	2276	208	0	2628.845068	1985.775061
1	1	retained	133	107	326.896596	1225	19	64	13715.920550	3160.472914
2	2	retained	114	95	135.522926	2651	0	0	3059.148818	1610.735904
3	3	retained	49	40	67.589221	15	322	7	913.591123	587.196542
4	4	retained	84	68	168.247020	1562	166	5	3950.202008	1219.555924
5	5	retained	113	103	279.544437	2637	0	0	901.238699	439.101397
6	6	retained	3	2	236.725314	360	185	18	5249.172828	726.577205
7	7	retained	39	35	176.072845	2999	0	0	7892.052468	2466.981741
8	8	retained	57	46	183.532018	424	0	26	2651.709764	1594.342984
9	9	churned	84	68	244.802115	2997	72	0	6043.460295	2341.838528

activity_days	driving_days	device
28	19	Android
13	11	iPhone
14	8	Android
7	3	iPhone
27	18	Android
15	11	iPhone
28	23	iPhone
22	20	iPhone
25	20	Android
7	3	iPhone

I identify the cells with missing values within the dataset.

```python
missing_values = df.isnull().sum()
print(missing_values)
```

output:

```
ID                         0
label                    700
sessions                   0
drives                     0
total_sessions             0
n_days_after_onboarding    0
total_navigations_fav1     0
total_navigations_fav2     0
driven_km_drives           0
duration_minutes_drives    0
activity_days              0
driving_days               0
device                     0
dtype: int64
```

The 'Label' column has 700 values equal to zero.

I check the variable type in different columns of the database:

input:

```python
tipi_colonne = df.dtypes

print(tipi_colonne)
```

output:

```
ID                         int64
label                      object
sessions                   int64
drives                     int64
total_sessions             float64
n_days_after_onboarding    int64
total_navigations_fav1     int64
total_navigations_fav2     int64
driven_km_drives           float64
duration_minutes_drives    float64
activity_days              int64
driving_days               int64
device                     object
dtype: object
```

Datatypes are object, int64, float64

I display the number of columns.

```
num_rows, num_columns = df.shape


print(f'Numero di righe: {num_rows}')
print(f'Numero di colonne: {num_columns}')
```

Numero di righe: 14999
Numero di colonne: 13

I isolate the null values in the 'Label' column and display their descriptive statistics as follows:

input

```
null_df = df[df['label'].isnull()]
null_df.describe()
print(null_df.describe())
```

output:

	ID	sessions	drives	total_sessions	n_days_after_onboarding	total_navigations_fav1	total_navigations_fav2	driven_km_drives	duration_minutes_drives	activity_days	driving_days
count	700	700	700	700	700	700	700	700	700	700	700
mean	7405,584286	80,83714286	67,79857143	198,4833479	1709,295714	118,7171429	30,37142857	3935,967029	1795,123358	15,38285714	12,12571429
std	4306,900234	79,98744031	65,27192596	140,5617147	1005,306562	156,3081399	46,30698444	2443,107121	1419,242246	8,772713768	7,626373292
min	77	0	0	5,582648005	16	0	0	290,1198107	66,58849334	0	0
25%	3744,5	23	20	94,05634032	869	4	0	2119,344818	779,0092713	8	6
50%	7443	56	47,5	177,2559249	1650,5	62,5	10	3421,156727	1414,966279	15	12
75%	11007	112,25	94	266,0580216	2508,75	169,25	43	5166,097373	2443,955404	23	18
max	14993	556	445	1076,879741	3498	1096	352	15135,39128	9746,253023	31	30

I isolate the values with non-null labels and extract their descriptive statistical characteristics."

input

```
not_null_values = df[~df['label'].isnull()]

# Display summary stats of rows without null values
not_null_values.describe()
print(not_null_values.describe())
```

ouput

	ID	sessions	drives	total_sessions	n_days_after_onboarding	total_navigations_fav1	total_navigations_fav2	driven_km_drives	duration_minutes_drives	activity_days	driving_days
count	14299	14299	14299	14299	14299	14299	14299	14299	14299	14299	14299
mean	7503,573117	80,62381985	67,25582209	189,5474085	1751,822505	121,7473949	29,63829638	4044,401535	1864,199794	15,54465347	12,18253025
std	4331,207621	80,73650152	65,94729528	136,1897643	1008,663834	147,7134284	45,35089044	2504,97797	1448,005047	9,016088256	7,83383523
min	0	0	0	0,2202109438	4	0	0	60,44125046	18,28208247	0	0
25%	3749,5	23	20	90,45773271	878,5	10	0	2217,31991	840,1813436	8	5
50%	7504	56	48	158,7185714	1749	71	9	3496,545617	1479,394387	16	12
75%	11257,5	111	93	253,5404499	2627,5	178	43	5299,972162	2466,928876	23	19
max	14998	743	596	1216,154633	3500	1236	415	21183,40189	15851,72716	31	30

I am checking for statistical significance in the differences between the two groups highlighted earlier through the two tables, especially the variables null_df and not_null_values. To do this, I am using the difference between the mean and standard deviation in the two tables and performing the Student's t-test on the means:

```
from scipy.stats import ttest_ind

# Colonne di interesse
colonne_di_interesse = ['sessions', 'drives',
'total_sessions', 'n_days_after_onboarding',
'total_navigations_fav1', 'total_navigations_fav2',
                        'driven_km_drives',
'duration_minutes_drives', 'activity_days', 'driving_days']

for colonna in colonne_di_interesse:
   # Estraggo media e deviazione standard dai due DataFrame
   media_not_null = not_null_values[colonna].mean()
   std_not_null = not_null_values[colonna].std()

   media_null = null_df[colonna].mean()
   std_null = null_df[colonna].std()

   # Calcolo la differenza nelle medie
   diff_media = media_not_null - media_null

   # Calcolo la differenza nelle deviazioni standard
   diff_std = std_not_null - std_null

   # Eseguo il test t di Student solo sulle medie
   t_statistic, p_value = ttest_ind(not_null_values[colonna],
null_df[colonna], nan_policy='omit')

   # Visualizzo i risultati
   print(f'Colonna: {colonna}')
   print(f'Differenza nelle Medie: {diff_media}')
   print(f'Differenza nelle Deviazioni Standard: {diff_std}')
   print(f'T-Statistic (sulle Medie): {t_statistic}')
   print(f'P-Value (sulle Medie): {p_value}')
   print('\n')
```

output:

Differenza nelle Medie: -0.2133230096010692
Differenza nelle Deviazioni Standard: 0.7490612128172529
T-Statistic (sulle Medie): -0.06828502759118525
P-Value (sulle Medie): 0.9455596529026835

Colonna: drives

Differenza nelle Medie: -0.5427493431109127
Differenza nelle Deviazioni Standard: 0.6753693216297592
T-Statistic (sulle Medie): -0.212705811704630002
P-Value (sulle Medie): 0.8315593243882341

Colonna: total_sessions
Differenza nelle Medie: -8.93593938320663
Differenza nelle Deviazioni Standard: -4.371950320851909
T-Statistic (sulle Medie): -1.692416324257919
P-Value (sulle Medie): 0.09058739410434061

Colonna: n_days_after_onboarding
Differenza nelle Medie: 42.5267907845704940
Differenza nelle Deviazioni Standard: 3.3572714591070962
T-Statistic (sulle Medie): 1.0893166327575272
P-Value (sulle Medie): 0.27603178398017114

Colonna: total_navigations_fav1
Differenza nelle Medie: 3.030252065579006
Differenza nelle Deviazioni Standard: -8.594711541532348
T-Statistic (sulle Medie): 0.5284705519018119
P-Value (sulle Medie): 0.5971806009666079

Colonna: total_navigations_fav2
Differenza nelle Medie: -0.7331321870660261
Differenza nelle Deviazioni Standard: -0.9560939939139459
T-Statistic (sulle Medie): -0.4171923629892723
P-Value (sulle Medie): 0.6765436825044607

Colonna: driven_km_drives
Differenza nelle Medie: 108.43450640776246
Differenza nelle Deviazioni Standard: 61.87084986955142
T-Statistic (sulle Medie): 1.119511731827141
P-Value (sulle Medie): 0.2629398181091204

Colonna: duration_minutes_drives
Differenza nelle Medie: 69.0764362808759
Differenza nelle Deviazioni Standard: 28.762801870909016

T-Statistic (sulle Medie): 1.2334711775343632
P-Value (sulle Medie): 0.21741935528374806

Colonna: activity_days
Differenza nelle Medie: 0.16179632941364552
Differenza nelle Deviazioni Standard: 0.2433744887352134
T-Statistic (sulle Medie): 0.4641527316178832
P-Value (sulle Medie): 0.6425450613837862

Colonna: driving_days
Differenza nelle Medie: 0.05681596115612386
Differenza nelle Deviazioni Standard: 0.2074619376702005
T-Statistic (sulle Medie): 0.18758419433997617
P-Value (sulle Medie): 0.85121051551152942

***Most of the differences** in means and standard deviations between the two groups (with empty label and with label) are not statistically significant. In general, a p-value greater than 0.05 suggests that there is not enough statistical evidence to reject the null hypothesis of no difference, indicating no significant difference.*

***The majority of p-values are above 0.05**, suggesting that there are no statistically significant differences in the means and standard deviations of the considered columns between the two groups.*

 # *Null values - device counts*

Count how many iPhone users had null values and how many Android users had null values using the function value_counts and then printing the result:

input:

```
device_counts = null_df['device'].value_counts()
print(device_counts)
```

output:

device
iPhone   447
Android   253

Name: count, dtype: int64

 # "I perform the same calculation but express the result in percentage. Input:"

```
device_percentages =
null_df['device_type'].value_counts(normalize=True) * 100

print(device_percentages)
```

output:

device
iPhone   63.857143
Android   36.142857
Name: proportion, dtype: float64

 # I perform the same calculation on the entire dataset and express the result in percentage.

```
full_device_percentages =
df['device'].value_counts(normalize=True)  * 100

print(full_device_percentages)
```

output:

device
iPhone   64.484299
Android   35.515701
Name: proportion, dtype: float64

Now I am going to examine the counts and percentages of users who churned vs. those who were retained in the **entire dataset:**

```
label_percentage1 = df['label'].value_counts()
label_percentage2 = df['label'].value_counts(normalize=True)  *
100

print(label_percentage1)
print(label_percentage2)
```

output:

retained   11763

churned   2536
Name: count, dtype: int64

label
retained   82.264494
churned   17.735506
Name: proportion, dtype: float64

The dataset contains 82% retained users and 18% churned users.
 # "Now I am going to compare the medians of each variable for churned and retained users, calculating the median to avoid that outliers excessively influence the results as it would happen if I used the mean:"

input:

```
# "I select the numerical columns within the dataset."
colonne_numeriche = df.select_dtypes(include=np.number)

# I group the DataFrame by the 'label' column.'
gruppi = df.groupby('label')

# I calculate the median for each group.
mediane_per_gruppo = gruppi[colonne_numeriche.columns].median()
```

output:

label	ID	sessions	drives	total_sessions	n_days_after_onboarding	total_navigations_fav1	total_navigations_fav2	driven_km_drives	duration_minutes_drives	activity_days	driving_days
churned	7477,5	59	50	164,3390421	1321	84,5	11	3652,655666	1607,183785	8	6
retained	7509	56	47	157,5867563	1843	68	9	3464,684614	1458,046141	17	14

The median allows me to observe the data by reducing the effect of outliers. In particular, **retained** users have fewer sessions, fewer drives, fewer total sessions, fewer kilometers, and fewer minutes of navigation. However, they have had more active days and more driving days. It appears that **churned** users have taken longer trips, covering more kilometers but with fewer active days and driving days.

 # Calculate the median kilometers per drive in the last month for both retained and churned users.

input:

```
colonne_numeriche = df.select_dtypes(include=np.number)

median_by_label =
colonne_numeriche.groupby(df['label']).median()

mediana_km_per_drive =
median_by_label['driven_km_drives']/median_by_label['drives']

print(mediana_km_per_drive)
```

output:

churned 73.053113
retained 73.716694
dtype: float64

The median user from both groups drove ~73 km/drive.

How many kilometers per driving day was this?

input:

```
mediana_per_driving_day =
median_by_label['driven_km_drives']/median_by_label['driving_days']
```

output:

Median result for kilometers per driving day

churned 608.775944
retained 247.477472

Now I calculate the median number of drives per driving day for each group

```
mediana_number_of_drives =
median_by_label['drives']/median_by_label['driving_days']

churned     8.333333
retained    3.357143
```

From the data, it is clear that churned users have covered many more kilometers per driving day compared to retained users, at a ratio of approximately 1 to 3. The same ratio applies to the number of rides per driving day. These are users who drive for many kilometers, indicating a different usage profile.

From these data, we can hypothesize that users who have abandoned the service are heavy users of the service, traveling extensively and covering many kilometers. Perhaps the app and the service do not meet the needs of these highly active users**.**

Finally, I am going to examine whether there is an imbalance in how many users churned by device type

input:

```
count_by_device = df.groupby(['label','device']).size()
print(count_by_device)
```

output:

```
label     device
churned   Android    891
          -     1645
retained  Android   4183
          iPhone    7580
```

I perform the same calculation but in percentage.

```
count_by_device_in_perc =
df.groupby('label')['device'].value_counts(normalize=True) * 100
print(count_by_device_in_perc)
```

Percentage output of the data:

```
label     device
churned   iPhone     64.865931
          Android    35.134069
retained  iPhone     64.439344
      Android    35.560656
```

The percentage of iPhone and Android users is very similar among both churned and retained users; therefore, no significant differences are noticeable.

1. **Did the data contain any missing values?** How many, and which variables were affected? Was there a pattern to the missing data?

The "Label" column has 700 *values equal to zero and is the only column with null values. The data suggests that there are no statistically significant differences in the means and standard deviations of the considered columns between the two groups, churned and retained.*
The smartphone operating system *data tells me that there are no significant differences between the groups with null and non-null values. The percentage of missing values for each device is similar to their representation in the overall data. There is nothing to suggest a non-random cause of the missing data.*

2. **What is a benefit** of using the median value of a sample instead of the mean?

The median is less affected by extreme values, making it a better choice when dealing with outliers. In general when data does not follow a normal distribution, the median is often a more appropriate indicator.

3. **Did your investigation give** rise to further questions that you would like to explore or ask the Waze team about?

If they have any suggestions or clues to explain the difference between churned and retained users regarding the median values of kilometers per drive and kilometers per driving day.

4. **What percentage** of the users in the dataset were Android users and what percentage were iPhone users?

"These are the percentage data for the entire dataset."
device
iPhone   64.484299
Android   35.515701

5. **What were some distinguishing** characteristics of users who churned vs. users who were retained?

From the data, it is clear that churned users have covered many more kilometers per driving day compared to retained users, at a ratio of approximately 1 to 3. The same ratio applies to the number of rides per driving day. These are users who drive for many kilometers, indicating a different usage profile.

From these data, we can hypothesize that users who have abandoned the service are heavy users of the service, traveling extensively and covering many kilometers. Perhaps the app and the service do not meet the needs of these highly active users**.**

6. **Was there an appreciable** difference in churn rate between iPhone users vs. Android users?

The percentage of iPhone and Android users is very similar among both churned and retained users; therefore, no significant differences are noticeable.