

# Exemplar\_Course 2 Waze project lab

January 7, 2024

## 1 Waze Project

### Course 2 - Get Started with Python

Welcome to the Waze Project!

Your Waze data analytics team is still in the early stages of their user churn project. Previously, you were asked to complete a project proposal by your supervisor, May Santner. You have received notice that your project proposal has been approved and that your team has been given access to Waze's user data. To get clear insights, the user data must be inspected and prepared for the upcoming process of exploratory data analysis (EDA).

A Python notebook has been prepared to guide you through this project. Answer the questions and create an executive summary for the Waze data team.

## 2 Course 2 End-of-course project: Inspect and analyze data

In this activity, you will examine data provided and prepare it for analysis. This activity will help ensure the information is,

1. Ready to answer questions and yield insights
2. Ready for visualizations
3. Ready for future hypothesis testing and statistical methods

**The purpose** of this project is to investigate and understand the data provided.

**The goal** is to use a dataframe constructed within Python, perform a cursory inspection of the provided dataset, and inform team members of your findings.

*This activity has three parts:*

**Part 1:** Understand the situation \* How can you best prepare to understand and organize the provided information?

**Part 2:** Understand the data

- Create a pandas dataframe for data learning, future exploratory data analysis (EDA), and statistical activities
- Compile summary information about the data to inform next steps

### **Part 3: Understand the variables**

- Use insights from your examination of the summary data to guide deeper investigation into variables

Follow the instructions and answer the following questions to complete the activity. Then, you will complete an Executive Summary using the questions listed on the PACE Strategy Document.

Be sure to complete this activity before moving on. The next course item will provide you with a completed exemplar to compare to your own work.

## **3 Identify data types and compile summary information**

### **4 PACE stages**

Throughout these project notebooks, you'll see references to the problem-solving framework PACE. The following notebook components are labeled with the respective PACE stage: Plan, Analyze, Construct, and Execute.

#### **4.1 PACE: Plan**

Consider the questions in your PACE Strategy Document and those below to craft your response:

##### **4.1.1 Task 1. Understand the situation**

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

*Begin by exploring your dataset and consider reviewing the Data Dictionary.*

**Answer:**

Prepare by reading in the data, viewing the data dictionary, and exploring the dataset to identify key variables for the stakeholder.

#### **4.2 PACE: Analyze**

Consider the questions in your PACE Strategy Document to reflect on the Analyze stage.

##### **4.2.1 Task 2a. Imports and data loading**

Start by importing the packages that you will need to load and explore the dataset. Make sure to use the following import statements:

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

```
[1]: # Import packages for data manipulation
import pandas as pd
import numpy as np
```

Then, load the dataset into a dataframe. Creating a dataframe will help you conduct data manipulation, exploratory data analysis (EDA), and statistical activities.

**Note:** As shown in this cell, the dataset has been automatically loaded in for you. You do not need to download the .csv file, or provide more code, in order to access the dataset and proceed with this lab. Please continue with this activity by completing the following instructions.

```
[2]: # Load dataset into dataframe
df = pd.read_csv('waze_dataset.csv')
```

#### 4.2.2 Task 2b. Summary information

View and inspect summary information about the dataframe by **coding the following**:

1. `df.head(10)`
2. `df.info()`

Consider the following questions:

1. When reviewing the `df.head()` output, are there any variables that have missing values?
2. When reviewing the `df.info()` output, what are the data types? How many rows and columns do you have?
3. Does the dataset have any missing values?

```
[3]: df.head(10)
```

```
[3]:   ID  label  sessions  drives  total_sessions  n_days_after_onboarding  \
0   0  retained      283    226      296.748273                2276
1   1  retained      133    107      326.896596                1225
2   2  retained      114     95      135.522926                2651
3   3  retained       49     40       67.589221                 15
4   4  retained       84     68      168.247020                1562
5   5  retained      113    103      279.544437                2637
6   6  retained        3      2      236.725314                 360
7   7  retained       39     35      176.072845                2999
8   8  retained       57     46      183.532018                 424
9   9  churned       84     68      244.802115                2997

      total_navigations_fav1  total_navigations_fav2  driven_km_drives  \
0                        208                        0      2628.845068
1                         19                       64      13715.920550
2                         0                        0      3059.148818
3                        322                        7       913.591123
4                        166                        5      3950.202008
```

5	0	0	901.238699
6	185	18	5249.172828
7	0	0	7892.052468
8	0	26	2651.709764
9	72	0	6043.460295

	duration_minutes_drives	activity_days	driving_days	device
0	1985.775061	28	19	Android
1	3160.472914	13	11	iPhone
2	1610.735904	14	8	Android
3	587.196542	7	3	iPhone
4	1219.555924	27	18	Android
5	439.101397	15	11	iPhone
6	726.577205	28	23	iPhone
7	2466.981741	22	20	iPhone
8	1594.342984	25	20	Android
9	2341.838528	7	3	iPhone

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 13 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   ID                                    14999 non-null  int64
1   label                                14299 non-null  object
2   sessions                             14999 non-null  int64
3   drives                               14999 non-null  int64
4   total_sessions                       14999 non-null  float64
5   n_days_after_onboarding              14999 non-null  int64
6   total_navigations_fav1              14999 non-null  int64
7   total_navigations_fav2              14999 non-null  int64
8   driven_km_drives                     14999 non-null  float64
9   duration_minutes_drives              14999 non-null  float64
10  activity_days                        14999 non-null  int64
11  driving_days                         14999 non-null  int64
12  device                               14999 non-null  object
dtypes: float64(3), int64(8), object(2)
memory usage: 1.5+ MB
```

### Answers:

1. None of the variables in the first 10 observations have missing values. Note that this does not imply the whole dataset does not have any missing values.
2. The variables `label` and `device` are of type `object`; `total_sessions`, `driven_km_drives`, and `duration_minutes_drives` are of type `float64`; the rest of the variables are of type `int64`. There are 14,999 rows and 13 columns.

3. The dataset has 700 missing values in the label column.

#### 4.2.3 Task 2c. Null values and summary statistics

Compare the summary statistics of the 700 rows that are missing labels with summary statistics of the rows that are not missing any values.

**Question:** Is there a discernible difference between the two populations?

```
[5]: # Isolate rows with null values
null_df = df[df['label'].isnull()]
# Display summary stats of rows with null values
null_df.describe()
```

```
[5]:
```

	ID	sessions	drives	total_sessions	\
count	700.000000	700.000000	700.000000	700.000000	
mean	7405.584286	80.837143	67.798571	198.483348	
std	4306.900234	79.987440	65.271926	140.561715	
min	77.000000	0.000000	0.000000	5.582648	
25%	3744.500000	23.000000	20.000000	94.056340	
50%	7443.000000	56.000000	47.500000	177.255925	
75%	11007.000000	112.250000	94.000000	266.058022	
max	14993.000000	556.000000	445.000000	1076.879741	

  

	n_days_after_onboarding	total_navigations_fav1	\
count	700.000000	700.000000	
mean	1709.295714	118.717143	
std	1005.306562	156.308140	
min	16.000000	0.000000	
25%	869.000000	4.000000	
50%	1650.500000	62.500000	
75%	2508.750000	169.250000	
max	3498.000000	1096.000000	

  

	total_navigations_fav2	driven_km_drives	duration_minutes_drives	\
count	700.000000	700.000000	700.000000	
mean	30.371429	3935.967029	1795.123358	
std	46.306984	2443.107121	1419.242246	
min	0.000000	290.119811	66.588493	
25%	0.000000	2119.344818	779.009271	
50%	10.000000	3421.156721	1414.966279	
75%	43.000000	5166.097373	2443.955404	
max	352.000000	15135.391280	9746.253023	

  

	activity_days	driving_days
count	700.000000	700.000000
mean	15.382857	12.125714

std	8.772714	7.626373
min	0.000000	0.000000
25%	8.000000	6.000000
50%	15.000000	12.000000
75%	23.000000	18.000000
max	31.000000	30.000000

```
[6]: # Isolate rows without null values
not_null_df = df[~df['label'].isnull()]
# Display summary stats of rows without null values
not_null_df.describe()
```

```
[6]:
```

	ID	sessions	drives	total_sessions \
count	14299.000000	14299.000000	14299.000000	14299.000000
mean	7503.573117	80.623820	67.255822	189.547409
std	4331.207621	80.736502	65.947295	136.189764
min	0.000000	0.000000	0.000000	0.220211
25%	3749.500000	23.000000	20.000000	90.457733
50%	7504.000000	56.000000	48.000000	158.718571
75%	11257.500000	111.000000	93.000000	253.540450
max	14998.000000	743.000000	596.000000	1216.154633

	n_days_after_onboarding	total_navigations_fav1 \
count	14299.000000	14299.000000
mean	1751.822505	121.747395
std	1008.663834	147.713428
min	4.000000	0.000000
25%	878.500000	10.000000
50%	1749.000000	71.000000
75%	2627.500000	178.000000
max	3500.000000	1236.000000

	total_navigations_fav2	driven_km_drives	duration_minutes_drives \
count	14299.000000	14299.000000	14299.000000
mean	29.638296	4044.401535	1864.199794
std	45.350890	2504.977970	1448.005047
min	0.000000	60.441250	18.282082
25%	0.000000	2217.319909	840.181344
50%	9.000000	3496.545617	1479.394387
75%	43.000000	5299.972162	2466.928876
max	415.000000	21183.401890	15851.727160

	activity_days	driving_days
count	14299.000000	14299.000000
mean	15.544653	12.182530
std	9.016088	7.833835
min	0.000000	0.000000

25%	8.000000	5.000000
50%	16.000000	12.000000
75%	23.000000	19.000000
max	31.000000	30.000000

#### Answer:

Comparing summary statistics of the observations with missing retention labels with those that aren't missing any values reveals nothing remarkable. The means and standard deviations are fairly consistent between the two groups.

#### 4.2.4 Task 2d. Null values - device counts

Next, check the two populations with respect to the `device` variable.

**Question:** How many iPhone users had null values and how many Android users had null values?

```
[7]: # Get count of null values by device
null_df['device'].value_counts()
```

```
[7]: iPhone      447
      Android    253
      Name: device, dtype: int64
```

**Answer:** > Of the 700 rows with null values, 447 were iPhone users and 253 were Android users.

Now, of the rows with null values, calculate the percentage with each device—Android and iPhone. You can do this directly with the `value_counts()` function.

```
[8]: # Calculate % of iPhone nulls and Android nulls
null_df['device'].value_counts(normalize=True)
```

```
[8]: iPhone      0.638571
      Android    0.361429
      Name: device, dtype: float64
```

How does this compare to the device ratio in the full dataset?

```
[9]: # Calculate % of iPhone users and Android users in full dataset
df['device'].value_counts(normalize=True)
```

```
[9]: iPhone      0.644843
      Android    0.355157
      Name: device, dtype: float64
```

The percentage of missing values by each device is consistent with their representation in the data overall.

There is nothing to suggest a non-random cause of the missing data.

Examine the counts and percentages of users who churned vs. those who were retained. How many of each group are represented in the data?

```
[10]: # Calculate counts of churned vs. retained
print(df['label'].value_counts())
print()
print(df['label'].value_counts(normalize=True))
```

```
retained    11763
churned      2536
Name: label, dtype: int64
```

```
retained    0.822645
churned      0.177355
Name: label, dtype: float64
```

This dataset contains 82% retained users and 18% churned users.

Next, compare the medians of each variable for churned and retained users. The reason for calculating the median and not the mean is that you don't want outliers to unduly affect the portrayal of a typical user. Notice, for example, that the maximum value in the `driven_km_drives` column is 21,183 km. That's more than half the circumference of the earth!

```
[11]: # Calculate median values of all columns for churned and retained users
df.groupby('label').median(numeric_only=True)
```

```
[11]:
```

	ID	sessions	drives	total_sessions	n_days_after_onboarding	\
label						
churned	7477.5	59.0	50.0	164.339042		1321.0
retained	7509.0	56.0	47.0	157.586756		1843.0

  

	total_navigations_fav1	total_navigations_fav2	driven_km_drives	\
label				
churned		84.5	11.0	3652.655666
retained		68.0	9.0	3464.684614

  

	duration_minutes_drives	activity_days	driving_days
label			
churned	1607.183785	8.0	6.0
retained	1458.046141	17.0	14.0

This offers an interesting snapshot of the two groups, churned vs. retained:

Users who churned averaged ~3 more drives in the last month than retained users, but retained users used the app on over twice as many days as churned users in the same time period.

The median churned user drove ~200 more kilometers and 2.5 more hours during the last month than the median retained user.

It seems that churned users had more drives in fewer days, and their trips were farther and longer in duration. Perhaps this is suggestive of a user profile. Continue exploring!



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

```
[12]: # Group data by `label` and calculate the medians
medians_by_label = df.groupby('label').median(numeric_only=True)
print('Median kilometers per drive:')
# Divide the median distance by median number of drives
medians_by_label['driven_km_drives'] / medians_by_label['drives']
```

Median kilometers per drive:

```
[12]: label
      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?

```
[13]: # Divide the median distance by median number of driving days
print('Median kilometers per driving day:')
medians_by_label['driven_km_drives'] / medians_by_label['driving_days']
```

Median kilometers per driving day:

```
[13]: label
      churned      608.775944
      retained      247.477472
      dtype: float64
```

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

```
[14]: # Divide the median number of drives by median number of driving days
print('Median drives per driving day:')
medians_by_label['drives'] / medians_by_label['driving_days']
```

Median drives per driving day:

```
[14]: label
      churned      8.333333
      retained      3.357143
      dtype: float64
```

The median user who churned drove 608 kilometers each day they drove last month, which is almost 250% the per-drive-day distance of retained users. The median churned user had a similarly disproportionate number of drives per drive day compared to retained users.

It is clear from these figures that, regardless of whether a user churned or not, the users represented in this data are serious drivers! It would probably be safe to assume that this data does not represent typical drivers at large. Perhaps the data—and in particular the sample of churned users—contains a high proportion of long-haul truckers.

In consideration of how much these users drive, it would be worthwhile to recommend to Waze that they gather more data on these super-drivers. It's possible that the reason for their driving so much is also the reason why the Waze app does not meet their specific set of needs, which may differ from the needs of a more typical driver, such as a commuter.

Finally, examine whether there is an imbalance in how many users churned by device type.

Begin by getting the overall counts of each device type for each group, churned and retained.

```
[15]: # For each label, calculate the number of Android users and iPhone users  
df.groupby(['label', 'device']).size()
```

```
[15]: label      device  
      churned   Android      891  
           iPhone    1645  
      retained   Android    4183  
           iPhone    7580  
      dtype: int64
```

Now, within each group, churned and retained, calculate what percent was Android and what percent was iPhone.

```
[16]: # For each label, calculate the percentage of Android users and iPhone users  
df.groupby('label')['device'].value_counts(normalize=True)
```

```
[16]: label      device  
      churned   iPhone    0.648659  
           Android    0.351341  
      retained   iPhone    0.644393  
           Android    0.355607  
      Name: device, dtype: float64
```

The ratio of iPhone users and Android users is consistent between the churned group and the retained group, and those ratios are both consistent with the ratio found in the overall dataset.

### 4.3 PACE: Construct

**Note:** The Construct stage does not apply to this workflow. The PACE framework can be adapted to fit the specific requirements of any project.

### 4.4 PACE: Execute

Consider the questions in your PACE Strategy Document and those below to craft your response:

#### 4.4.1 Task 3. Conclusion

Recall that your supervisor, May Santer, asked you to share your findings with the data team in an executive summary. Consider the following questions as you prepare to write your summary. Think about key points you may want to share with the team, and what information is most relevant to the user churn project.

##### Questions:

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

*The dataset has 700 missing values in the `label` column. There was no obvious pattern to the missing values.*

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

*Mean is subject to the influence of outliers, while the median represents the middle value of the distribution regardless of any outlying values.*

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

*Yes. For example, the median user who churned drove 608 kilometers each day they drove last month, which is almost 250% the per-drive-day distance of retained users. It would be helpful to know how this data was collected and if it represents a non-random sample of users.*

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

*Android users comprised approximately 36% of the sample, while iPhone users made up about 64%.*

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

*Generally, users who churned drove farther and longer in fewer days than retained users. They also used the app about half as many times as retained users over the same period.*

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

*No. The churn rate for both iPhone and Android users was within one percentage point of each other. There is nothing suggestive of churn being correlated with device.*

**Congratulations!** You've completed this lab. However, you may not notice a green check mark next to this item on Coursera's platform. Please continue your progress regardless of the check mark. Just click on the "save" icon at the top of this notebook to ensure your work has been logged.