Compare Ad Effectiveness Using A/B Tests with Python Project

Task Group 1 - Import, Inspect, and Merge

Task 1

Import the **pandas** and **numpy** libraries using their standard aliases.

In [1]:
```
import numpy as np
import pandas as pd
```

Task 2

Import the CSV file `users.csv` and assign it to the variable `users`. Preview the first five rows.

In [3]:
```
users = pd.read_csv('users.csv')
users.head()
```

Out[3]:

	user_id	timestamp	device_id	clicked
0	U790620	2022-01-01 00:23:33 Saturday	M003	False
1	U584867	2022-01-01 01:30:07 Saturday	M001	False
2	U128681	2022-01-01 01:30:14 Saturday	M002	True
3	U694898	2022-01-01 01:31:55 Saturday	M003	False
4	U456823	2022-01-01 03:18:25 Saturday	M001	False

▶ *What is the structure of this dataset? Toggle to check!*

Task 3

Import the CSV file `advertisements.csv` and assign it to the variable `advertisements`. Preview the first five rows.

In [5]:
```
advertisements = pd.read_csv('advertisements.csv')
advertisements.head()
```

Out[5]:

	user_id	timestamp	ad_source	ad_version
0	U790620	2022-01-01 00:23:33 Saturday	Twitter	A
1	U584867	2022-01-01 01:30:07 Saturday	Google	B
2	U128681	2022-01-01 01:30:14 Saturday	TikTok	B
3	U694898	2022-01-01 01:31:55 Saturday	TikTok	A
4	U456823	2022-01-01 03:18:25 Saturday	Google	A

▶ *What is the structure of this dataset? Toggle to check!*

Task 4

There are two `user_id` columns, one in each DataFrame. Let's do a quick check to see if they have the same number of unique users. If they don't, we'll know to be careful about this issue later on in our analysis.

Count the unique number of `user_id` s in `users` and separately in `advertisements`.

Print both counts.

In [10]:
```
print('Users unique user_id:', users['user_id'].nunique(), '\nAdvertiseements unique user_id:',
advertisements['user_id'].nunique())
```

```
Users unique user_id: 15122
Advertiseements unique user_id: 14602
```

Task 5

In order to effectively analyze the results of our A/B test, we'll need rows that contain user information from both DataFrames. Specifically, we'll need to know whether or not a user `clicked` on the ad in `users` and we'll need to know which `ad_version` the user viewed in `advertisements`.

Merge `users` and `advertisements` on the `user_id` and `timestamp` columns. Make sure to only return rows that have data from both DataFrames.

Save the merged DataFrame to a variable named `users_ads` and preview the first five rows.

In [13]:
```
users_ads = pd.merge(left=users, right=advertisements, left_on=['user_id', 'timestamp'], right_on=['user_id', 'time
users_ads.head()
```

Out[13]:

	user_id	timestamp	device_id	clicked	ad_source	ad_version
0	U790620	2022-01-01 00:23:33 Saturday	M003	False	Twitter	A
1	U584867	2022-01-01 01:30:07 Saturday	M001	False	Google	B
2	U128681	2022-01-01 01:30:14 Saturday	M002	True	TikTok	B
3	U694898	2022-01-01 01:31:55 Saturday	M003	False	TikTok	A
4	U456823	2022-01-01 03:18:25 Saturday	M001	False	Google	A

▶ *Why do we use a merge that returns matching rows from both DataFrames? Toggle to check!*

Task Group 2 - Use Aggregations to Calculate Click-Through Rate

Task 6

To begin our exploration, let's count how many times each `ad_version` was viewed by users in `users_ads`.

Group `users_ads` by `ad_version` and `count` the `user_id` column. Save the result to the variable `ad_view_count`, and rename the `user_id` column `num_views`.

Preview the final result.

In [16]:
```
ad_view_count = users_ads.groupby('ad_version').agg({'user_id':'count'}).rename(columns={'user_id':'num_views'})
ad_view_count
```

Out[16]:

ad_version	num_views
A	7154
B	7270

Task 7

There may be instances where individual users saw the ads more than once. Let's re-do the groupby from Task 6, but adding a column with the number of unique `user_id`s for each ad.

Modify `ad_view_count` to also return the count of unique users in `user_id`. Name this new column `nunique`.

Preview the modified `ad_view_count`.

In [19]:
```
ad_view_count = users_ads.groupby('ad_version').agg({'user_id':['count', 'nunique']}).rename(columns={'user_id':'',
ad_view_count
```

Out[19]:

ad_version	num_views	nunique
A	7154	7125
B	7270	7232

▶ *How should we address users who saw ads more than once? Toggle to check!*

Task 8

Group by `ad_version` and compute the **percentage** of users who clicked on each ad. This metric is known as the **click-through rate (CTR)** and is widely used as a performance metric.

Save the resultin DataFrame to the variable `ad_ctr_pct`.

Rename the `clicked` column to `click_rate`.

Preview `ad_ctr_pct`.

In [22]:
```
ad_ctr_pct = users_ads.groupby('ad_version').agg({'clicked':'mean'}).rename(columns={'clicked':'click_rate'})
ad_ctr_pct
```

Out[22]:

ad_version	click_rate
A	0.124126
B	0.194223

▶ *What did we learn about each ad version? Toggle to check!*

Task Group 3: Compare Ad Performances by Social Media Platform

Task 9

Let's break down click-through rate by social media platform.

Group `users_ads` by `ad_source` and `ad_version`, and compute the percent of `True` values in `clicked`.

Rename the column containing the percents `ctr` and view the full resulting DataFrame.

In [25]:
```
ad_social = users_ads.groupby(['ad_source', 'ad_version']).agg({'clicked': 'mean'}).rename(columns={'clicked':'ctr'
ad_social
```

Out[25]:

ad_source	ad_version	ctr
Google	A	0.128385
	B	0.198825
Meta	A	0.129664
	B	0.188034
TikTok	A	0.115819
	B	0.202358
Twitter	A	0.119624
	B	0.175070

Task 10

It is a little hard to read the long-format table from Task 9.

Pivot `users_ads` to create a wide-format version of this table, and name it `ad_social`.

Print all rows of `ad_social`.

In [26]:
```
ad_social = pd.pivot_table(
    ad_social,
    index = 'ad_source',
    columns='ad_version',

)
ad_social
```

Out[26]:

	ctr	
ad_version	A	B
ad_source		
Google	0.128385	0.198825
Meta	0.129664	0.188034
TikTok	0.115819	0.202358
Twitter	0.119624	0.175070

▶ *What did we learn about the ad performances across various social media platforms? Toggle to check!*

Task Group 4 - Compare Ad Performances by Tech Device

Task 11

Let's now investigate the click-through rates broken down by device (cell phone, tablet, etc.)

Import the CSV file `devices.csv` and assign it to the variable `devices`. Preview the full dataset.

How many different devices are there?

In [27]:
```
devices = pd.read_csv('devices.csv')
devices
```

Out[27]:

	device_id	device_type	brand
0	M001	Mobile	Apple
1	M002	Mobile	Samsung
2	M003	Mobile	Google
3	M004	Mobile	Huawei
4	M005	Mobile	Xiaomi
5	M006	Mobile	vivo
6	P001	PC	Apple
7	P002	PC	Dell
8	P003	PC	HP
9	P004	PC	ASUS
10	P005	PC	Lenovo
11	P006	PC	Other
12	S001	Smartwatch	Apple
13	S002	Smartwatch	Samsung
14	S003	Smartwatch	FitBit
15	ST01	Streaming	Roku
16	ST02	Streaming	Apple TV
17	ST03	Streaming	Gaming Console
18	T001	Tablet	Apple
19	T002	Tablet	Samsung
20	T003	Tablet	Amazon
21	T004	Tablet	Microsoft

▶ *What is the structure of this dataset? Toggle to check!*

Task 12

Let's now combine each user's ad information with their device information.

Merge `users_ads` with `devices` using the `device_id` column. Make sure to return all of the rows in `users_ads`, along with matching device information if it exists.

Name the merged DataFrame `users_devices` and preview the first five rows.

In [29]:
```
users_devices = pd.merge(left=users_ads, right=devices, left_on='device_id', right_on='device_id', how='outer')
users_devices.head()
```

Out[29]:

	user_id	timestamp	device_id	clicked	ad_source	ad_version	device_type	brand
0	U790620	2022-01-01 00:23:33 Saturday	M003	False	Twitter	A	Mobile	Google
1	U694898	2022-01-01 01:31:55 Saturday	M003	False	TikTok	A	Mobile	Google
2	U447163	2022-01-01 06:37:08 Saturday	M003	True	Google	B	Mobile	Google
3	U143188	2022-01-01 12:36:02 Saturday	M003	False	TikTok	B	Mobile	Google
4	U533059	2022-01-02 10:08:20 Sunday	M003	False	Meta	B	Mobile	Google

Task 13

Calculate the percentage of users who clicked on the advertisement based on their `device_type` and `ad_version` they viewed.

In [41]:
```
users_devices.groupby(['device_type', 'ad_version']).agg({'clicked': 'mean'})
```

Out[41]:

		clicked
device_type	**ad_version**	
Mobile	**A**	0.121057
	B	0.215877
PC	**A**	0.121311
	B	0.182425
Tablet	**A**	0.128315
	B	0.171348

▶ *What did we learn about the ad performances across tech devices? Toggle to check!*

Task Group 5 - Weekday and Weekend Performance by Device Type

Task 14

Let's break our analysis down further by weekday versus weekend user behavior.

There are various approaches we could use to explore this question, but let's practice using the **split-apply-combine (SAC)** technique.

Recall, SAC involves a three-step process where we:

1. **Split** the dataset into one or more pieces
2. **Apply** a function or transformation to each piece
3. **Combine** the results from each piece

We've provided starter code below that creates a new column `day_of_week` extracted from the `timestamp` column indicating the day on which the user viewed the ad.

Count the number of users who viewed each ad, grouped by the day of the week and the ad version they saw.

In [42]:
```
# Create 'day_of_week' column
users_devices['day_of_week'] = users_devices['timestamp'].str.split(' ', expand=True)[2]

## YOUR SOLUTION HERE
users_devices.groupby(['day_of_week', 'ad_version']).agg({'clicked': 'count'})
```

Out[42]:

		clicked
day_of_week	**ad_version**	
Friday	**A**	1061
	B	1061
Monday	**A**	1014
	B	1037
Saturday	**A**	1045
	B	996
Sunday	**A**	979
	B	1057
Thursday	**A**	1018
	B	997
Tuesday	**A**	991
	B	1068
Wednesday	**A**	1046
	B	1054

Task 15

Split `users_devices` into two DataFrames:

- `weekends` filtering for users who saw either ad on Saturday or Sunday

- `weekdays` filtering for users who saw either ad from Monday to Friday

Use the `|` operator, and see the hint for more of a refresher on Boolean masks!

Preview the first five rows in `weekdays`.

In [48]:
```
weekend = (users_devices['day_of_week'] == 'Saturday')|(users_devices['day_of_week'] == 'Sunday')
```

In [49]:
```
weekday = ~weekend
users_devices[weekday]
```

Out[49]:

	user_id	timestamp	device_id	clicked	ad_source	ad_version	device_type	brand	day_of_week
7	U484678	2022-01-03 00:10:18 Monday	M003	False	Meta	A	Mobile	Google	Monday
8	U597754	2022-01-03 00:53:23 Monday	M003	False	TikTok	B	Mobile	Google	Monday
9	U695803	2022-01-03 04:17:55 Monday	M003	False	TikTok	A	Mobile	Google	Monday
10	U331858	2022-01-04 05:18:44 Tuesday	M003	False	TikTok	B	Mobile	Google	Tuesday
11	U128142	2022-01-04 07:32:42 Tuesday	M003	True	Google	B	Mobile	Google	Tuesday
...	...	...	...	...	...	...	...	...	...
14427	NaN	NaN	S002	NaN	NaN	NaN	Smartwatch	Samsung	NaN
14428	NaN	NaN	S003	NaN	NaN	NaN	Smartwatch	FitBit	NaN
14429	NaN	NaN	ST01	NaN	NaN	NaN	Streaming	Roku	NaN
14430	NaN	NaN	ST02	NaN	NaN	NaN	Streaming	Apple TV	NaN
14431	NaN	NaN	ST03	NaN	NaN	NaN	Streaming	Gaming Console	NaN

10355 rows × 9 columns

Task 16

Apply aggregation functions.

Create a new DataFrame `weekday_ctr` that computes

- the total number of views on weekdays
- the percent of clicks on weekdays

grouped by `device_type` and `ad_version`.

Name the two columns `weekday_views` and `weekday_rate`, and then reset the index of the DataFrame `weekday_ctr`.

Print the full `weekday_ctr` DataFrame.

In [50]:
```
weekday_ctr = users_devices[weekdays].groupby(['device_type', 'ad_version']).agg({'clicked':['count', 'mean']})
weekday_ctr = weekday_ctr.rename(columns={'clicked':'', 'count':'weekday_views', 'mean':'weekday_rate'})
weekday_ctr
```

Out[50]:

		weekday_views	weekday_rate
device_type	**ad_version**		
Mobile	**A**	2517	0.125944
	B	2594	0.212799
PC	**A**	1309	0.128342
	B	1285	0.189105
Tablet	**A**	1009	0.142716
	B	1012	0.164032

Task 17

Create a new DataFrame `weekend_ctr` that computes

- the total number of clicks on weekends
- the percent of clicks on weekends

grouped by `device_type` and `ad_version`.

Name the two columns `weekend_views` and `weekend_rate`, and then reset the index of the DataFrame `weekend_ctr`.

Print the full `weekend_ctr` DataFrame.

In [51]:
```
weekend_ctr = users_devices[weekend].groupby(['device_type', 'ad_version']).agg({'clicked':['count', 'mean']})
weekend_ctr = weekend_ctr.rename(columns={'clicked':'', 'count':'weekend_views', 'mean':'weekend_rate'})
weekend_ctr
```

Out[51]:

		weekend_views	weekend_rate
device_type	**ad_version**		
Mobile	**A**	1002	0.108782
	B	996	0.223896
PC	**A**	521	0.103647
	B	513	0.165692
Tablet	**A**	386	0.090674
	B	412	0.189320

Task 18

Combine `weekday_ctr` and `weekend_ctr` by merging on the `device_type` and `ad_version` columns into a DataFrame named `combined_ctr`. Make sure to return matching rows from both DataFrames.

Preview all of the click-through rates in `combined_ctr`.

In [55]:
```
combined_ctr = pd.merge(left=weekday_ctr, right=weekend_ctr, left_on=['device_type', 'ad_version'], right_on=['devi
combined_ctr
```

Out[55]:

		weekday_views	weekday_rate	weekend_views	weekend_rate
device_type	**ad_version**				
Mobile	**A**	2517	0.125944	1002	0.108782
	B	2594	0.212799	996	0.223896
PC	**A**	1309	0.128342	521	0.103647
	B	1285	0.189105	513	0.165692
Tablet	**A**	1009	0.142716	386	0.090674
	B	1012	0.164032	412	0.189320