Bike-Share Case Study

This report provides the results and step-by-step explanation of the data analysis performed for a bike-sharing case study. The data belongs to a bike-sharing company that has two kinds of users: annual members and casual riders. The goal of the study was to identify how annual members and casual riders use the bikes differently in order to help the stake-holders decide whether or not to target converting casual riders into annual members in the next marketing campaign. Python was used to perform the analysis using data collected from January-November 2023 and downloaded from https://divvy-tripdata.s3.amazonaws.com/index.html.

Dataset exploration & cleaning:

The code that was used to perform the data exploration can be found in the Jupyter Notebook cleaning.ipynb. The main functions are explained below:

- *read_data*:
  Here the .csv file for the bike rides of each month is read and stored into a dictionary called "data". Each element in the dictionary has a key (the name of the month) and a value (the panada dataframe that holds the data entries). This simplifies the access of the entries for each corresponding month, by using the month as the key (e.g. data["May"] retrieves the dataframe that holds the entries from May). In Figure 1 we can see the first and last 5 entries of bike rides from May:

	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	start_lat	start_lng	end_lat	end_lng	member_casual
0	0D9FA920C3062031	electric_bike	2023-05-07 19:53:48	2023-05-07 19:58:32	Southport Ave & Belmont Ave	13229	NaN	NaN	41.939408	-87.663831	41.930000	-87.650000	member
1	92485E5FB5888ACD	electric_bike	2023-05-06 18:54:08	2023-05-06 19:03:35	Southport Ave & Belmont Ave	13229	NaN	NaN	41.939482	-87.663848	41.940000	-87.690000	member
2	FB144B3FC8300187	electric_bike	2023-05-21 00:40:21	2023-05-21 00:44:36	Halsted St & 21st St	13162	NaN	NaN	41.853793	-87.646719	41.860000	-87.650000	member
3	DDEB93BC2CE9AA77	classic_bike	2023-05-10 16:47:01	2023-05-10 16:59:52	Carpenter St & Huron St	13196	Damen Ave & Cortland St	13133	41.894556	-87.653449	41.915983	-87.677335	member
4	C07B70172FC92F59	classic_bike	2023-05-09 18:30:34	2023-05-09 18:39:28	Southport Ave & Clark St	TA1308000047	Southport Ave & Belmont Ave	13229	41.957081	-87.664199	41.939478	-87.663748	member
...	...	...	...	...	...	...	...	...	...	...	...	...	...
604822	48BDA26F34445546	electric_bike	2023-05-18 10:26:43	2023-05-18 10:48:00	Clark St & Elmdale Ave	KA1504000148	NaN	NaN	41.990876	-87.669721	42.000000	-87.660000	member
604823	573025E5EDE10DE1	electric_bike	2023-05-17 14:32:48	2023-05-17 14:45:37	State St & 33rd St	13216	NaN	NaN	41.834734	-87.625798	41.830000	-87.620000	member
604824	D88D48898C6FB63E	electric_bike	2023-05-17 07:59:29	2023-05-17 08:04:54	Columbus Dr & Randolph St	13263	NaN	NaN	41.884422	-87.619393	41.880000	-87.630000	member
604825	4692DCD2F87497F5	electric_bike	2023-05-18 08:34:48	2023-05-18 08:38:40	Public Rack - Karlov Ave & Lawrence Ave	1127.0	NaN	NaN	41.970000	-87.730000	41.970000	-87.740000	member
604826	6ACB7E383473D019	electric_bike	2023-05-29 21:16:58	2023-05-29 21:24:35	State St & 33rd St	13216	NaN	NaN	41.834715	-87.625764	41.840000	-87.650000	member

Figure 1: First and last 5 entries of bike rides from May (data["May"])

From the entries we can see that the data consists of 13 columns: 1) ride id, 2) the type of bike, 3-4) date and time for the start and end of the ride, 5-12) the name, id, latitude and longitude of the start and end stations, and finally 13) whether the user was a casual rider or a member. In order to explore the dataset a bit further, the number of unique values for each column in data["May"] is calculated and shown in Figure 2:

Column Name:	NUnique
ride_id	604827
rideable_type	3
started_at	503683
ended_at	505259
start_station_name	1287
start_station_id	1250
end_station_name	1254
end_station_id	1210
start_lat	188591
start_lng	185410
end_lat	4759
end_lng	4762
member_casual	2

Figure 2: No. of unique values for every column in data["May"]

As is expected, the columns *rideable_type* and *member_casual* have a small number of unique values, whereas the rest of the columns do not. The unique values in these two columns are:

  - *rideable_type*: [electric_bike, classic_bike, docked_bike]
  - *member_casual*: [member, casual]

- *count_entries*:
  After the csv files are read into the dataframes, the method *count_entries* is used to collect further information about the dataset. It finds the number of entries per file as well as the number of columns. This is done in order to check that the data format is consistent across the different files. Next, the method calculates the total number of bike rides in the entire dataset. There is an option within the method to remove duplicates. Therefore, the method is first called with the remove duplicates option deactivated, in order to get a preliminary feel of the dataset. And then the method is called again with the remove duplicates option activated. The results are then written to output files which are shown in Figure 3.

Original_BikeRides

Month	No Of Entries	No Of Cols
January	190301	13
February	190445	13
March	258678	13
April	426590	13
May	604827	13
June	719618	13
July	767650	13
August	771693	13
September	666371	13
October	537113	13
November	362518	13
Total:	5495804	
Average:	499618	

BikeRides_without_Duplicates

Month	No Of Entries	No Of Cols
January	190301	13
February	190445	13
March	258678	13
April	426590	13
May	604827	13
June	719618	13
July	767650	13
August	771693	13
September	666371	13
October	537113	13
November	362518	13
Total:	5495804	
Average:	499618	

Figure 3: No of entries and columns before and after removing duplicates

On the left is the result of running the method without removing duplicates, and on the right is the result after removing duplicates. We can see that all the files have the same number of columns, which is a good preliminary indicator of the consistency of data across the months. In total the dataset contains almost 5.5 Million entries, with an average of approximately 500,000 entries per month. From January to March the number of rides is relatively lower than the average, which is expected as these are cold months. This is confirmed by the peak in the number of rides during the Summer months June to August. The number of entries before and after removing duplicates is identical, therefore the original dataset did not have any duplicates.

- *check_NAN*:
  Given that a brief look at the entries from May already showed a couple of NaN values (Figure 1 *end_station_name*, and *end_station_id*), the method *check_NAN* calculates the percentage of NaN values for each column and month. The results are shown in Figure 4. As we can see the columns *start_station_name*, *start_station_id*, *end_station_name*, *end_station_id* in every month have 13-17% null values. The columns *end_lat* and *end_long* have less than 1% null values.

- *clean_data*:
  After exploring the dataset, we can see that the extractable information can be divided into information about the:

  1. rider (casual/member)
  2. bike (electric/classic/docked)
  3. ride (length, date, location)

  Since, the goal of the analysis is to find out how casual riders differ from members, it seems that the all information is relevant to the analysis, with the exception of the

NaN_Percentages

Month	ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	start_lat	start_lng	end_lat	end_lng	member_casual
January	0	0	0	0	14 %	14 %	14 %	14 %	0	0	< 1%	< 1%	0
February	0	0	0	0	13 %	13 %	14 %	14 %	0	0	< 1%	< 1%	0
March	0	0	0	0	13 %	13 %	14 %	14 %	0	0	< 1%	< 1%	0
April	0	0	0	0	14 %	14 %	16 %	16 %	0	0	< 1%	< 1%	0
May	0	0	0	0	14 %	14 %	15 %	15 %	0	0	< 1%	< 1%	0
June	0	0	0	0	16 %	16 %	17 %	17 %	0	0	< 1%	< 1%	0
July	0	0	0	0	16 %	16 %	16 %	16 %	0	0	< 1%	< 1%	0
August	0	0	0	0	15 %	15 %	16 %	16 %	0	0	< 1%	< 1%	0
September	0	0	0	0	15 %	15 %	16 %	16 %	0	0	< 1%	< 1%	0
October	0	0	0	0	15 %	15 %	16 %	16 %	0	0	< 1%	< 1%	0
November	0	0	0	0	15 %	15 %	15 %	15 %	0	0	< 1%	< 1%	0

Figure 4: Percentage of null values for each column across the various months

ride location. This geographical location would have been important if for example the goal of the analysis was to find out whether more stations should be added and where to do so. Therefore, since the locations seem to be irrelevant and contain null values, it is safer to drop these columns rather than drop the entries that contain null values. The column *ride_id* also does not provide any valuable information for the current analysis. Thus, the relevant columns needed from this point onwards are the: *rideable_type*, *started_at*, *ended_at*, *member_casual*. So the first task performed by the method *clean_data* is to drop the columns that are no longer needed.

Next, is data formatting. We have already looked at the columns *rideable_type* and *member_casual*, and saw that they have the expected values. As for the columns *started_at* and *ended_at*, a quick check shows that these columns are stored as strings of characters. So first, they are converted into a datetime format. Second, we need to check whether the *ended_at* time always comes after *started_at* time. By adding a column *started_before_ended* which is True if the ride time ended before it started, and False otherwise. We can then filter using this column to see when it is True. The result for the month of May is shown in Figure 5.

	rideable_type	started_at	ended_at	member_casual	ended_before_started
8308	classic_bike	2023-05-29 17:34:21	2023-05-29 17:34:09	member	True
38552	electric_bike	2023-05-29 16:57:34	2023-05-29 16:57:27	casual	True
103546	electric_bike	2023-05-26 15:39:47	2023-05-26 15:38:17	member	True
103547	electric_bike	2023-05-26 15:38:53	2023-05-26 15:38:17	member	True
209340	classic_bike	2023-05-07 15:54:58	2023-05-07 15:54:47	casual	True
211708	classic_bike	2023-05-23 17:39:38	2023-05-23 17:39:35	casual	True
216859	classic_bike	2023-05-13 18:08:15	2023-05-13 18:08:09	member	True
336480	electric_bike	2023-05-29 11:31:41	2023-05-29 11:31:33	member	True
417351	classic_bike	2023-05-27 05:31:51	2023-05-27 05:31:37	member	True
456170	electric_bike	2023-05-30 07:40:55	2023-05-30 07:39:58	member	True

Figure 5: Entries in May when the ended_at time is before the started_at time

Looking at the entries in Figure 5, we can see that indeed there are cases when the *ended_at* time is before the *started_at* time. It can be that in these incidents the start and end time were switched due to some glitch, perhaps the bike rental time being only a few seconds (shorter than the server response time). In the entire dataset of approximately 5.5 Million entries, there is a total of 262 entries that have this issue. Since, the dataset is large, we can simply drop these entries.

Finally, after dropping the columns related to location, the entries with the switched times, the new dataset is stored as *cleaned_data*. In order to ensure that the new dataset has the correct shape, the method *count_entries* is called once again and used to compare the cleaned dataset to the original one. The result is shown in Figure 6.

Original_BikeRides

Month	No Of Entries	No Of Cols
January	190301	13
February	190445	13
March	258678	13
April	426590	13
May	604827	13
June	719618	13
July	767650	13
August	771693	13
September	666371	13
October	537113	13
November	362518	13
Total:	5495804	
Average:	499618	

BikeRides_Cleaned

Month	No Of Entries	No Of Cols
January	190301	4
February	190445	4
March	258678	4
April	426590	4
May	604827	4
June	719618	4
July	767650	4
August	771693	4
September	666371	4
October	537113	4
November	362518	4
Total:	5495804	
Average:	499618	

Figure 6: No of entries and columns before and after dropping the null values

As we can see only the number of columns has changed (from 13 to 4), and the number of entries remains the same. As a final check, the method *check_NAN* is applied to the cleaned data, the result is as expected and is shown in Figure 8:

NaN_Percentages_Clean

Month	rideable_type	started_at	ended_at	member_casual
January	0	0	0	0
February	0	0	0	0
March	0	0	0	0
April	0	0	0	0
May	0	0	0	0
June	0	0	0	0
July	0	0	0	0
August	0	0	0	0
September	0	0	0	0
October	0	0	0	0
November	0	0	0	0

Figure 7: Percentage of null values after cleaning

- *prepare_data*:

Now that the data is clean and in the correct format, we can extract the required information for analysis. The method *prepare_data* adds two new columns: 1. *ride_length*: the difference between the columns *ended_at* and *started_at*, 2. *day_of_week*: extracted from the date in *started_at*. Figure **??** shows the first and last 5 entries of the data["May"] after cleaning and preparation.

	rideable_type	started_at	ended_at	member_casual	ride_length	day_of_week
0	electric_bike	2023-05-07 19:53:48	2023-05-07 19:58:32	member	0 days 00:04:44	Sunday
1	electric_bike	2023-05-06 18:54:08	2023-05-06 19:03:35	member	0 days 00:09:27	Saturday
2	electric_bike	2023-05-21 00:40:21	2023-05-21 00:44:36	member	0 days 00:04:15	Sunday
3	classic_bike	2023-05-10 16:47:01	2023-05-10 16:59:52	member	0 days 00:12:51	Wednesday
4	classic_bike	2023-05-09 18:30:34	2023-05-09 18:39:28	member	0 days 00:08:54	Tuesday
...	...	...	...	...	...	...
604822	electric_bike	2023-05-18 10:26:43	2023-05-18 10:48:00	member	0 days 00:21:17	Thursday
604823	electric_bike	2023-05-17 14:32:48	2023-05-17 14:45:37	member	0 days 00:12:49	Wednesday
604824	electric_bike	2023-05-17 07:59:29	2023-05-17 08:04:54	member	0 days 00:05:25	Wednesday
604825	electric_bike	2023-05-18 08:34:48	2023-05-18 08:38:40	member	0 days 00:03:52	Thursday
604826	electric_bike	2023-05-29 21:16:58	2023-05-29 21:24:35	member	0 days 00:07:37	Monday

Figure 8: First and last 5 entries of bike rides from May after cleaning and preparation

Analysis: