# Bike-Share Case Study

This report provides the results as well as the 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 case-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. The data used in this case-study was from January-November 2023. Each month's data was stored in a csv file, and was downloaded from https://divvy-tripdata.s3.amazonaws.com/index.html.

## Data-Set 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 csv entries). This simplifies the access of the entries for each corresponding month, by using the month as the key (e.g. data["February"] retrieves the dataframe that holds the entries from February). Below 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 |

From the entries of data["May"], we can see that the data for May 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 pandas method *nunique()* is used to return the number of unique values for each column in data["May"]. The output is shown below.

```
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
```

As should be expected, the only columns that have unique values are *rideable_type* and *member_casual*. The panadas method *unique()* is then be used to find these values:

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

• *count_entries*:
After the dataset is read into the dataframes, the method *count_entries* collects 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, how big it is, how the entires varies across the months. And then the method is called again with the remove duplicates option activated. The results are then written to output files which are shown below.

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 | |

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. 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, this method calculates the percentage of NaN values. The number of null values for each column is calculated using the pandas function *isna()*. The percentage of null values for each column and month is shown below. As we can see the columns *start_station_name*, *start_station_id*, *end_station_name*, *end_station_id* in every month have around 13-17% null values. The columns *end_lat* and *end_long* have less than 1% null values.

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 |

- *drop_NANs*:

Now that the percentage of null values has been identified, the next step is to determine how to deal with these null values. As previously mentioned, the goal of the analysis is to find out how casual riders differ from members. After exploring the dataset, it is clear that this information can be obtained by looking at how the ride lengths vary as well as the type of bikes used. The ride length can be determined either by looking at the time or the distance. Since the information related to the distance is harder to process as well as the fact that it contains null values, it is safer to rely on the time. Therefore, the relevant columns needed from this point onwards are the: *ride_id*, *rideable_type*, *started_at*, *ended_at*, *member_casual*. Which means that we can drop all the other columns which contain the null values. Therefore, the method *drop_NANs* drops the columns related to start and end stations and writes the data into a new dictionary called *cleaned_data*. After dropping the start and end stations, we can call the method *count_entries* once again, but this time passing along the *cleaned_data* in order to compare the cleaned dataset shape to the original one:

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_NaNsRemoved

| Month | No Of Entries | No Of Cols |
|---|---|---|
| January | 190301 | 5 |
| February | 190445 | 5 |
| March | 258678 | 5 |
| April | 426590 | 5 |
| May | 604827 | 5 |
| June | 719618 | 5 |
| July | 767650 | 5 |
| August | 771693 | 5 |
| September | 666371 | 5 |
| October | 537113 | 5 |
| November | 362518 | 5 |
| Total: | 5495804 | |
| Average: | 499618 | |

As expected only the number of columns has changed (from 13 to 5), and the number of entries remains the same as the original. One final check, we apply the *check_NAN* method to the cleaned data, the result is as expected and shown is below:

NaN_Percentages_Clean

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