Report on Datasets and Processing

Housing Prices in Germany

🏠 **Apartment rental offers in Germany** dataset from Kaggle:
https://www.kaggle.com/datasets/corrieaar/apartment-rental-offers-in-germany

The dataset contains rental offers scraped from Germany's biggest real estate online platform - Immoscout24. This dataset only contains offers for rental properties. While it was created from 2018 to 2019, it is still relevant and gives a good idea of the housing situation in different parts of Germany.

Pre-processing

The strategy for preprocessing is the following:

1. If a variable contains a lot of missing values:
    - We remove it if it doesn't have much relevance for this analysis
    - We keep it to display if it is relevant at any point
    - There is no need to perform imputation and handle all missing values, since the goal is not to build a model but to conduct a meaningful exploratory analysis of this dataset.
2. Remove irrelevant variables:
    - Some of the variables are out of the scope of this analysis
    - "Description" and "Facilities" columns have already been dropped to lower the size of the dataset
3. Remove duplicate variables
    - There are variables with a different name, which contains the same data in the same or worse format. These columns should be dropped.

Airbnb rent prices in European cities

This analysis is based on several independent datasets that share the same structure.

The datasets used are available here:

- https://zenodo.org/records/4446043

For this analysis, only data has been taken for workdays for Barcelona, Amsterdam, Berlin, and Budapest.

The columns are as following:

- `realSum`: the full price of accommodation for two people and two nights in EUR
- `room_type`: the type of the accommodation
- `room_shared`: dummy variable for shared rooms
- `room_private`: dummy variable for private rooms
- `person_capacity`: the maximum number of guests
- `host_is_superhost`: dummy variable for superhost status
- `multi`: dummy variable if the listing belongs to hosts with 2-4 offers
- `biz`: dummy variable if the listing belongs to hosts with more than 4 offers

- `cleanliness_rating`: cleanliness rating
- `guest_satisfaction_overall`: overall rating of the listing
- `bedrooms`: number of bedrooms (0 for studios)
- `dist`: distance from city centre in km
- `metro_dist`: distance from nearest metro station in km
- `attr_index`: attraction index of the listing location
- `attr_index_norm`: normalised attraction index (0-100)
- `rest_index`: restaurant index of the listing location
- `attr_index_norm`: normalised restaurant index (0-100)
- `lng`: longitude of the listing location
- `lat`: latitude of the listing location

Pre-processing & Processing

The `preprocess_dataset` function is responsible for cleaning up the dataset by removing unnecessary columns.

1. Column Removal: The function uses the DataFrame.drop method to remove the following columns from the dataset:

Preprocessing in `visualize_dataset`

Room Type Filtering

The function allows users to filter the dataset by room_type using a dropdown (`st.selectbox`). If a specific room type is selected (other than "All"), the dataset is filtered to include only rows where the room_type matches the selected value:

```
if room_type_filter and room_type_filter != "All":
    df = df[df["room_type"] == room_type_filter]
```

Color Metric Calculation:

A new column, `color_metric`, is added to the dataset.

This column is calculated based on the selected `color_metric_col` (e.g., `realSum` or `guest_satisfaction_overall`). The color_metric column is created by applying a lambda function to scale the values of the selected column into RGBA color values:

```
df["color_metric"] = df[color_metric_col].apply(
    lambda x: (
        int(255 / df[color_metric_col].max() * x),  # Red
        100,                                        # Green
        150,                                        # Blue
        255                                         # Transparency
    )
)
```

Renaming Columns for Mapping:

The dataset's longitude (lng) and latitude (lat) columns are renamed to lon and lat, respectively, to match the requirements of `st.map`:

```
df.rename(columns={"lng": "lon", "lat": "lat"})
```