Impact of Forest Coverage on Urban Air Quality: A Global Analysis

Author: Avsar Vora

Introduction

In this data science project, I aim to explore the impact of forest coverage on urban air quality across different regions of the world. The primary question is: **"How does forest coverage impact urban air quality across different regions of the world?"**

Datasources

Datasource1: World Forest Area

- Metadata URL: https://www.kaggle.com/datasets/webdevbadger/world-forest-area/data
- Data URL: https://www.kaggle.com/datasets/webdevbadger/world-forest-area/download
- Data Source & Data Type: Kaggle - CSV
- License: Creative Commons Attribution 4.0 International License.

The dataset comprises 34 columns, encompassing detailed information such as the country name, country code, and the percentage of forest area as a proportion of the total land area. The dataset spans from 1990 to 2021, with individual columns representing the annual percentage of forest area for each year within this period.

Datasource2: World Air Quality Index

- Metadata URL: https://www.kaggle.com/datasets/adityaramachandran27/world-air-quality-index-by-city-and-coordinates/data
- Data URL: https://www.kaggle.com/datasets/adityaramachandran27/world-air-quality-index-by-city-and-coordinates/download
- Data Source & Data Type: Kaggle - CSV
- License: Creative Commons Attribution 4.0 International License.

This dataset contains detailed information on various countries, including the number of cities, their geographic coordinates (latitude and longitude), and different air quality indices recorded in 2021.

Libraries Used

- **Pandas**: To create and manipulate data frames
- **SQLAlchemy**: To dump data frames into sqlite database
- **kaggle**: To retreive data from Kaggle and use Api keys to access data

```
%pip install --upgrade pip
%pip install pandas==2.2.2
%pip install SQLAlchemy==2.0.30
%pip install kaggle==1.6.14
```

Constant values to for each data frames: dataset_path, file_name, db_name, sqlalchemy_datatype

```
# Dataset const values to generate download & process data
DATASET = {
    "WORLD_FOREST_DATA": {
        "dataset_path": "webdevbadger/world-forest-area",
        "file_name": "forest_area_percent.csv",
        "db_name": "world_forest_data",
        "sqlalchemy_datatype": {
            "Country Name": TEXT,
            "1990": FLOAT(asdecimal=True),
            "1991": FLOAT(asdecimal=True),
            . # Due to the limit of pages in the report only showing the structure of the columns
            .
            .
            "2021": FLOAT(asdecimal=True)
        }
    },
    "WORLD_AIR_QUALITY_DATA": {
        "dataset_path": "adityaramachandran27/world-air-quality-index-by-city-and-coordinates",
        "file_name": "AQI and Lat Long of Countries.csv",
        "db_name": "world_air_quality_data",
        "sqlalchemy_datatype": {
            "Country": TEXT,
            "AQI Value": FLOAT(asdecimal=True),
            "CO AQI Value": FLOAT(asdecimal=True),
            "Ozone AQI Value": FLOAT(asdecimal=True),
            "NO2 AQI Value": FLOAT(asdecimal=True),
            "PM2.5 AQI Value": FLOAT(asdecimal=True)
        }
    },
}
```

Data Cleaning (Forest Dataset)

The dataset was downloaded and required minimal cleaning due to its high quality. The only modification made was the removal of one column containing **country code**.

```
def get_and_preprocess_forest_dataframe():
    # create forest dataframe from csv
    forest_df = pd.read_csv(DATASET["WORLD_FOREST_DATA"]["file_name"])

    # preprocess steps for forest dataframe
    forest_df = forest_df.drop(columns=['Country Code'])

    return forest_df
```

Data Cleaning (World AQI Dataset)

The dataset was downloaded and required minimal cleaning due to its high quality. The columns dropped for analysis are specified in the code as **columns**, while other columns, such as country and AQI values, were retained for analysis.

```
def get_and_preprocess_air_quality_dataframe():
    # create air quality dataframe from csv
    air_quality_df = pd.read_csv(DATASET["WORLD_AIR_QUALITY_DATA"]["file_name"])

    # preprocess steps for air quality dataframe
    air_quality_df = air_quality_df.drop(
        columns=['City', 'AQI Category', 'CO AQI Category', 'Ozone AQI Category', 'NO2 AQI Category',
                 'PM2.5 AQI Category', 'lat', 'lng'])

    return air_quality_df


def dump_dataset_to_db(dataframe, db_name, datatype):
    db_engine = create_engine(f"sqlite:///../data/{db_name}.sqlite")
    dataframe.to_sql(db_name, db_engine, index=False, if_exists='replace', dtype=datatype)
    return
```

Final Result

The output data of the pipeline consists of two SQLite tables: **world_air_quality_data.sqlite** and **world_forest_data.sqlite**, both derived from original CSV input files. The data structure is well-organized and clean, with minimal preprocessing required due to the high quality of the initial datasets. SQLite was chosen as the output format for its efficiency in handling structured data and ease of integration with various analytical tools. However, potential issues for the final report could include the need for further data normalization and addressing any unforeseen inconsistencies or biases in the original datasets that may affect analysis results.

```
def data_collector():
    # insert forest data into sqlite database
    dump_dataset_to_db(dataframe=forest_df, db_name=DATASET['WORLD_FOREST_DATA']['db_name'],
                       datatype=DATASET['WORLD_FOREST_DATA']['sqlalchemy_datatype'])

    # insert air quality data into sqlite database
    dump_dataset_to_db(dataframe=air_quality_df, db_name=DATASET['WORLD_AIR_QUALITY_DATA']['db_name'],
                       datatype=DATASET['WORLD_AIR_QUALITY_DATA']['sqlalchemy_datatype'])

    return
```

+ Code + Text