Week 3: Data Understanding & documentation.

Objective: In this lab session, you will learn basic data exploration techniques and documentation, including metadata. We will use Google Colab - Python IDE for most of our tasks.

Go to https://colab.research.google.com/ to start a new Python notebook. You will need to log in to use the Google Colab. You can use your university email or your personal.

To start, make sure you download all the .csv files from Blackboard.

```
#import all the useful libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from dataprep.eda import plot
from dataprep.eda import plot_correlation
from dataprep.eda import plot_missing
```

Exercise 1. Ingest Population data (they are in CSV formats)

```
#ingest the projects data set and do basic wrangling
pop = pd.read_csv('/content/Data/population_data.csv', skiprows=4)
#replace with the right file path
```

Exercise 2a. Print the top rows of the Population data

```
# print the first few rows from the top of the population data
pop.head()
```

Exercise 2b. Print the last rows of the Population data

```
# print the last few rows from the population data
pop.___()
```

Exercise 3a. Drop some columns from the Population data

```
#drop the unnamed62 and country code column in the data
pop.drop(['Unnamed: 62'], inplace=True, axis=1)
```

Exercise 3b. Print the Basic information of the Population data

```
#check the basic info around the data types
pop.info()
```

Exercise 4. Investigate in % the number of missing values in each column for the data

```
# Run this code cell to check how many null values are in the data
```

```
# complete the code cell to check how many null values are in the pop
missingness_pop= ______.isnull().sum() / len(pop) * 100
missingness_pop.head(50)
```

Exercise 5. Visualise the missingness for the data

```
# Calculate the percentage of missing values for each column
missing_percentage = (population.isnull().sum() / len(population)) *
100

# Create a bar plot of the missing values
plt.figure(figsize=(12, 6))
sns.barplot(x=missing_percentage.index, y=missing_percentage.values)
plt.xticks(rotation=90)
plt.xlabel('Columns')
plt.ylabel('Percentage of Missing Values')
plt.title('Missing Values in Population Data')
plt.show()
```

Exercise 6a. Investigate the descriptive statistics of the data

```
#plot the descriptive statistics of population data, remember we are
using pop as the name of the variable
pop.describe().T
```

By default, the describe method gives the descriptive statistics for numeric variables. However, we can modify it to provide descriptive statistics for categorical variables, but the statistics would not be the same as you have learned from lectures. See next exercise.

Exercise 6b. Investigate the descriptive statistics of the data for categorical variables

```
# descriptive statistics for categorical variables
pop.describe(include='O').T
```

Exercise 7. Line plot of the United Kingdom population data over the years

```
# Create a line plot
# prepare the data for visualisation, you may replace the country name
#to others if you want to try for others. However, make sure the name
#of the country is exactly written as the one in the file.

country_name = 'United Kingdom'

# Filter the data for the chosen country
country_data = pop[pop['Country Name'] == country_name]

# Select the relevant columns (years and population values)
years = country_data.columns[4:]  #years start from the 4th column
pop_values = country_data[years].values[0]
# Create a line plot
```

```
plt.figure(figsize=(12, 6))
plt.plot(years, population_values)
plt.title(f'Population Trend for {country_name}')
plt.xlabel('Year')
plt.ylabel('Population')
plt.grid(True)
plt.xticks(rotation=45, ha='right')
plt.show()
```

Exercise 8a. Outlier detection in the population data for the year 2016 using boxplot

```
# Extract population data for the year 2016
population_2016 = pop['2016']

# Create a box plot to visualize outliers
plt.figure(figsize=(10, 6))
sns.boxplot(x=population_2016)
plt.title('Box Plot of Population in 2016')
plt.xlabel('Population in 2016')
plt.show()
```

Exercise 8b. Outlier detection in the population data for the year 2016 using IQR

```
# Identify outliers using the IQR method
Q1 = population_2016.quantile(0.25)
Q3 = population_2016.quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = population_2016[(population_2016 < lower_bound) |
(population_2016 > upper_bound)]

# Print the countries with outlier population in 2016
print("Countries with outlier population in 2016:")
for country in population[population_2016.isin(outliers)]['Country
Name']:
  print(country)
```

Exercise 8c. Outlier detection in the population data for the year 2016 using Scatterplot

```
# Create a scatter plot to visualize the relationship between
population and outliers in 2016
plt.figure(figsize=(10, 6))
plt.scatter(population['Country Name'], population_2016)
plt.title('Scatter Plot of Population in 2016')
plt.xlabel('Country Name')
plt.ylabel('Population in 2016')
plt.xticks(rotation=90)
```

```
plt.show()
```

From the three methods, we can see many outliers due to regional data getting aggregated together. We can remove these data points and redo the analysis. Country names like the world, south Asia, and upper-middle-income, among others, must be removed.

Exercise 8d. Outlier detection in the population data for the year 2016 by removing non-country/aggregates values

```
# create a list of all non-country names
non_country=['Arab World', 'East Asia & Pacific (excluding high
income)', 'Early-demographic dividend', 'East Asia & Pacific', 'Europe
& Central Asia (excluding high income)', 'Europe & Central Asia', 'Euro
area', 'European Union', 'Fragile and conflict affected situations',
'High income', 'Heavily indebted poor countries (HIPC)', 'IBRD only',
'IDA & IBRD total', 'IDA total', 'IDA blend', 'Indonesia', 'IDA only',
'Latin America & Caribbean (excluding high income)', 'Latin America &
Caribbean', 'Least developed countries: UN classification', 'Low
income', 'Lower middle income', 'Low & middle income', 'Late-
demographic dividend', 'Middle East & North Africa', 'Middle income',
'Middle East & North Africa (excluding high income)', 'North America',
'OECD members',  'Pre-demographic dividend', 'Post-demographic
dividend', 'South Asia', 'Sub-Saharan Africa (excluding high income)',
'Sub-Saharan Africa', 'East Asia & Pacific (IDA & IBRD countries)',
'Europe & Central Asia (IDA & IBRD countries)', 'Latin America & the
Caribbean (IDA & IBRD countries)', 'Middle East & North Africa (IDA &
IBRD countries)', 'South Asia (IDA & IBRD)', 'Sub-Saharan Africa (IDA &
IBRD countries)', 'Upper middle income', 'World']

# Remove rows where 'Country Name' is in the non_country list
population_filtered = pop[~pop['Country Name'].isin(non_country)]

# Extract population data for the year 2016 from the filtered data
population_2016_filtered = population_filtered['2016']
```

i. Using boxplot

```
# Create a box plot to visualize outliers after removing non-country
names
plt.figure(figsize=(10, 6))
sns.boxplot(x=population_2016_filtered)
plt.title('Box Plot of Population in 2016 (After Removing Non-Country
Names)')
plt.xlabel('Population in 2016')
plt.show()
```

ii. Using IQR

```
# Identify outliers using the IQR method (after removing non-country
names)
Q1_filtered = population_2016_filtered.quantile(0.25)
Q3_filtered = population_2016_filtered.quantile(0.75)
IQR_filtered = Q3_filtered - Q1_filtered
```

```
lower_bound_filtered = Q1_filtered - 1.5 * IQR_filtered
upper_bound_filtered = Q3_filtered + 1.5 * IQR_filtered

outliers_filtered = population_2016_filtered[
    (population_2016_filtered < lower_bound_filtered) |
(population_2016_filtered > upper_bound_filtered)
]

# Print the countries with outlier population in 2016 (after removing
non-country names)
print("Countries with outlier population in 2016 (After Removing Non-
Country Names):")
for country in
population_filtered[population_2016_filtered.isin(outliers_filtered)]['
Country Name']:
    print(country)
```

iii. Using Scatter plot

```
# Create a scatter plot to visualize the relationship between
population and outliers in 2016 (after removing non-country names)
plt.figure(figsize=(10, 6))
plt.scatter(population_filtered['Country Name'],
population_2016_filtered)
plt.title('Scatter Plot of Population in 2016 (After Removing Non-
Country Names)')
plt.xlabel('Country Name')
plt.ylabel('Population in 2016')
plt.xticks(rotation=90)
plt.show()
```

Extra Exercises:

1. **Using tools to carry out quick data exploration**

There are many tools/libraries for quick EDA. Below are the names of a few of them in Python.

- Pandas Profiling
- DataPrep

Both of them can produce:

- Descriptive statistics: mean, median, standard deviation, quartiles, etc.
- Missing values: percentage of missing values per column
- Data types: identification of numerical, categorical, and text columns
- Histograms and correlations: visual representation of data distribution and relationships between variables
- Outlier detection: identification of potential outliers in the data
- Interactive visualisations.

For the lab session, we will use the Data Prep. Kindly explore Pandas profiling at your own time https://docs.profiling.ydata.ai/latest/

2. **Plot the EDA for the population data:**

```
plot(pop) #eda using plot method from dataprep
```

3. **Investigate the correlation among variables of the population data**

```
plot_correlation(population)
```

4. **Investigate outliers in the population for the years 2015 and 2010.**

Follow what was learnt in the Lab session to investigate the outliers in the population data for 2015 and 2010.

5. **Write a Python script to ingest data from the World Bank via API**

```
url =
'http://api.worldbank.org/v2/countries/br;cn;us;de/indicators/SP.POP.TOTL/?format=json&per_page=10000'
```

i. Restructure the data into rows and columns, and afterwards, write the output as a .csv
ii. Use Dataprep or Pandas profiling to explore the saved .csv quickly.

6. **Write good documentation for all the procedures in this lab session, including the extra exercises.**