Department of Computer Engineering
University of Peradeniya

**CO 544 Machine Learning and Data Mining**
**Lab 04**

February 24, 2022

## 1. Objective

Provide students hands-on experience in the Matplotlib, Seaborn, and Pandas Python modules.

## 2. Introduction

Making plots and visualizations are one of the most important tasks in data mining and machine learning. It may be a part of the exploratory process; for example, identifying outliers, data transformations, or coming up with ideas for models. Matplotlib is a Python package, which provides a wide variety of plot types such as lines, bars, pie charts, and histograms.

## 3. Data Visualization

1. Anatomy of a figure

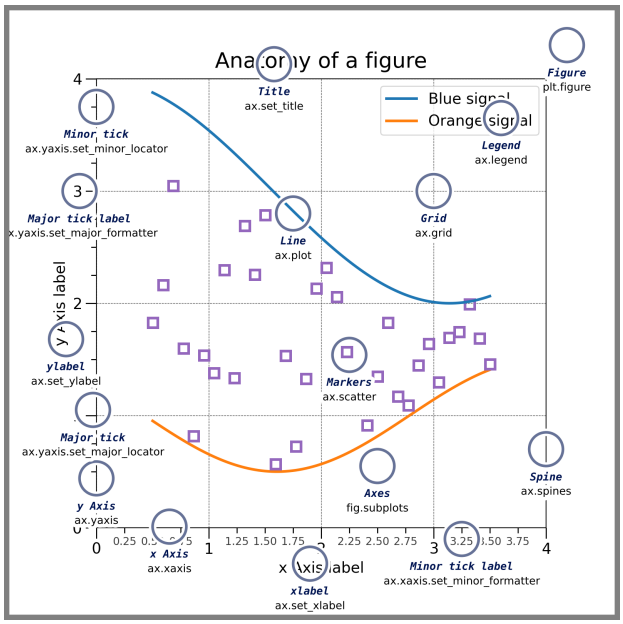


Figure 1: Anatomy of a figure (source: https://matplotlib.org/)

2. Importing libraries and the dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In this labsheet, the Gapminder dataset is used for visualization purposes. This dataset contains several key demographic and economic statistics for many countries, across many years. For more information, you can check the Gapminder repository.

```
gapminder_df = pd.read_csv('gapminder.tsv', sep='\t') #Read the dataset
gapminder_df.head() #Display first few rows
```

Let's explore the statistics for the most recent year in the dataframe and create a new dataframe to store them.

```
recent_year = gapminder_df['year'].max() #Get the most recent year
recent_year_df = gapminder_df[gapminder_df['year'] == recent_year] #Filter the
                                        dataframe based on the most recent year
recent_year_df.shape #Shape of the new dataframe
```

The descriptive summary statistics of a dataframe can be generated using the `pandas.describe()` function. This includes mean, count, std deviation, percentiles, and min-max values of all the features. Let's get an idea of how per-capita GDP (i.e., `gdpPercap` feature) was distributed across all of the countries during the most recent year.

```
recent_year_df['gdpPercap'].describe()
```

3. Histograms

   A histogram is a plot that lets you discover, and show, the underlying frequency distribution of a set of continuous data. This allows the inspection of the data for its underlying distribution (e.g., normal distribution), outliers, skewness, etc. and visualizes how many of the data points are in each of bins, each of which has a pre-defined range.

   Let's create a histogram for the per-capita GDP in the most recent year.

```
plt.figure(figsize=(10, 6)) #Create a figure
plt.hist(recent_year_df['gdpPercap']); #Plot the histogram
```

   To make this histogram more interpretable we can use several Matplotlib elements composing a figure as given in Figure 1.

```
plt.hist(recent_year_df['gdpPercap'])
plt.title('Distribution of Global Per-Capita GDP in 2007')
plt.xlabel('Per-Capita GDP (In dollars)')
plt.ylabel('Number of countries');
```

   Each bar in the histogram represents a bin. The height of the line represents the number of countries within the range of values spanned by the bin. In the plot shown in Figure 2, we used the default number of bins (i.e. 10), now let's use more bins by specifying the `bin=20` parameter.

```
plt.hist(recent_year_df['gdpPercap'], bins=20)
plt.title('Distribution of Global Per-Capita GDP in 2007')
plt.xlabel('Per-Capita GDP (In dollars)')
plt.ylabel('Number of countries');
```

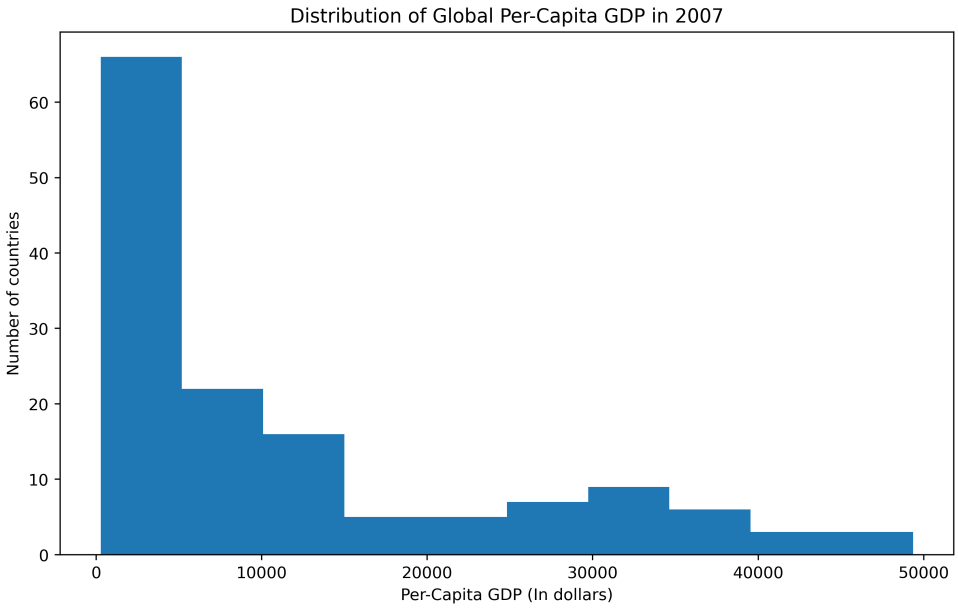


Figure 2: Distribution of the per-capita GDP in 2007

---

**TODO 1:** Describe the distribution of data in histograms for different bin sizes. Briefly explain any mechanism for calculating the optimal number of bins in a histogram.

---

4. Bar plots

   Moreover, it would be useful to get a perception on how many countries per continent in the dataset. To do that, let's create a new dataframe that includes only unique combinations of country and continent.

```
countries_df = gapminder_df[['country', 'continent']] #New dataframe with
                                                       selected features
countries_df = countries_df.drop_duplicates() #Drop duplicate combinations
countries_df.shape
```

   To get the number of countries per continent, the `groupby()` method can be used. The count of unique counties in each continent is then tallied. Here, we use the `as_index=False` so that the continent name gets its own column, and is not used as the index.

```
country_counts_df = countries_df.groupby('continent', as_index=False).agg('count')
country_counts_df.head()
```

   Let us use a bar plot, which plots numerical values for levels of a categorical feature as bars, to plot the data.

```
no_continents = len(country_counts_df)
x = range(no_continents)
print(x)

y = country_counts['country']

continents = country_counts_df['continent']
print(continents)
```

```
plt.figure(figsize=(10, 6))
plt.bar(x, y) #Plot bar graph
plt.xlabel('Continent')
plt.ylabel('Number of Countries');
plt.title('Number of Countries per Continent')
plt.xticks(x, continents);
```

5. Box plots

   Since we already have an intuition on how GDP was distributed in 2007, and how many countries are in each continent, we can explore how GDP is distributed within each continent. Here, instead of plotting separate histograms for each continent, we can take advantage of box plots, which is especially designed for such purposes.

```
continent_gdp_recent = []
for c in continents:
    cur_cont = recent_year_df[recent_year_df['continent'] == c] #New dataframe for
                                                        current continent
    cur_gdp_vals = cur_cont['gdpPercap'].values #Store Per-capita GDP values of
                                          current continent in an array
    continent_gdp_recent.append(cur_gdp_vals) #Append values to the list
```

   we can then use the list of arrays (i.e., `continent_gdp_recent`) to make a boxplot.

```
plt.figure(figsize=(10, 6))
plt.boxplot(continent_gdp_recent)
plt.title('Per-Capita GDP Distributions Over Continent')
plt.xlabel('Continent')
plt.ylabel('Per-Capita GDP (In dollars)')
plt.xticks(range(1, len(continents) + 1), continents);
```

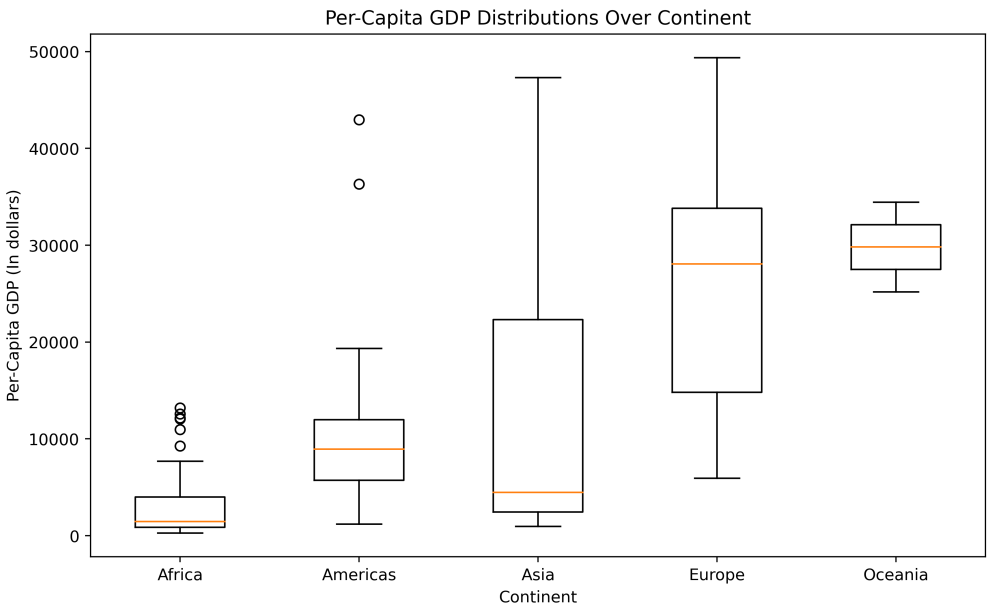


Figure 3: Distribution of Per-Capita GDP Over Continent

In boxplots, a box is created for each array, where the top and bottom lines of the box indicate the third and first quartiles, respectively. An orange bar in the middle indicates the median.

"Whiskers" show the extent of the extreme values. "Fliers" or "outliers" are shown in black circles and are drawn for all values that are outside of the whiskers, or interquartile value.

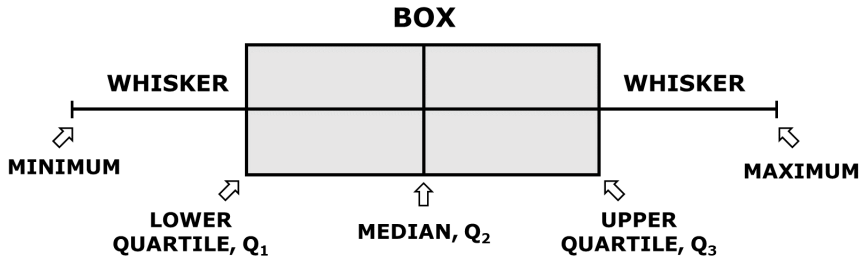


Figure 4: Overview of a box and whisker plot (source: https://www.statcan.gc.ca/)

6. Line plots

Line plots show how a continuous variable changes over time. In general, the variable that measures time is plotted on the x-axis and the continuous variable is plotted on the y-axis. Here, a line plot can be employed to visualize the change in the per capita GDP of a single country over time. Let us examine how the per capita GDP of Sri Lanka has changed over the years.

```
lk = gapminder_df[gapminder_df['country'] == 'Sri Lanka']

plt.figure(figsize=(10, 6))
plt.plot(lk['year'], lk['gdpPercap'])
plt.title('Per-capita GDP of Sri Lanka')
plt.xlabel('Year')
plt.ylabel('Per-capita GPD (In dollars)')
plt.grid();
```

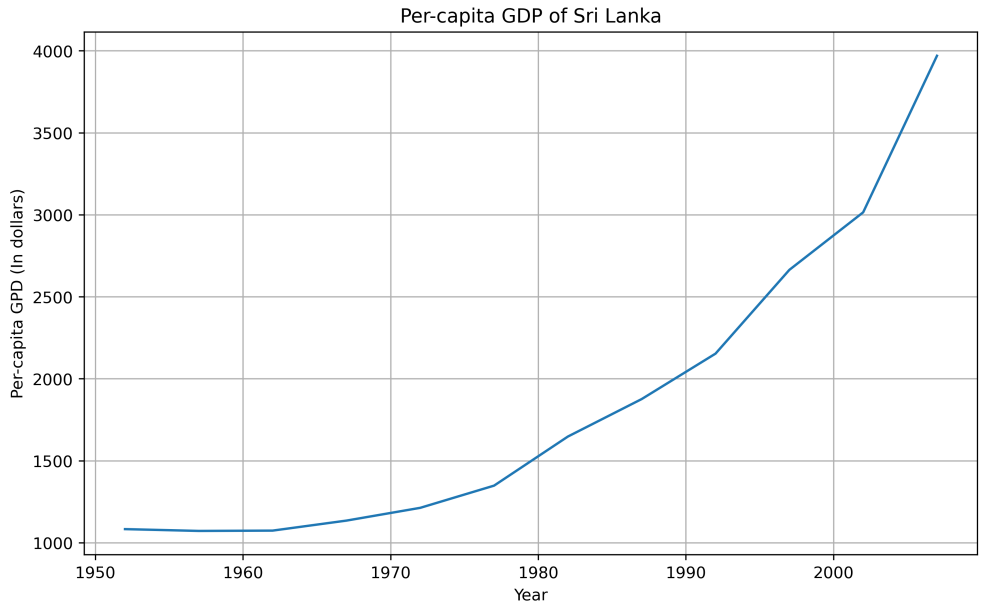


Figure 5: Changing of Per-capita GDP of Sri Lanka over time

---

**TODO 2:** Compare how Sri Lanka's per capita GDP has changed relative to its neighboring country India by plotting a second line plot on the same graph.

Hint: Distinguish lines using a legend (an area describing the elements of a graph.)

---

7. Scatter plots

   Suppose we have to observe the correlation between two variables, where both variables can have the same value several times, unlike time-series data where the time variable is constantly increasing. A scatter plot is a powerful way to visualize the joint distribution of data points across two variables.

   Let us visualize the relationship between the per capita GDP (`gdpPercap` on the x-axis) and life expectancy (`lifeExp` the y-axis) across all countries over the entire period.

```
plt.figure(figsize=(10, 6))
plt.scatter(gapminder_df['gdpPercap'], gapminder_df['lifeExp'], marker='+')
plt.title('Correlation between the Per-Capita GDP and Life Expectancy')
plt.xlabel('Per-Capita GDP (In Dollars)')
plt.ylabel('Life Expectancy (In years)');
```

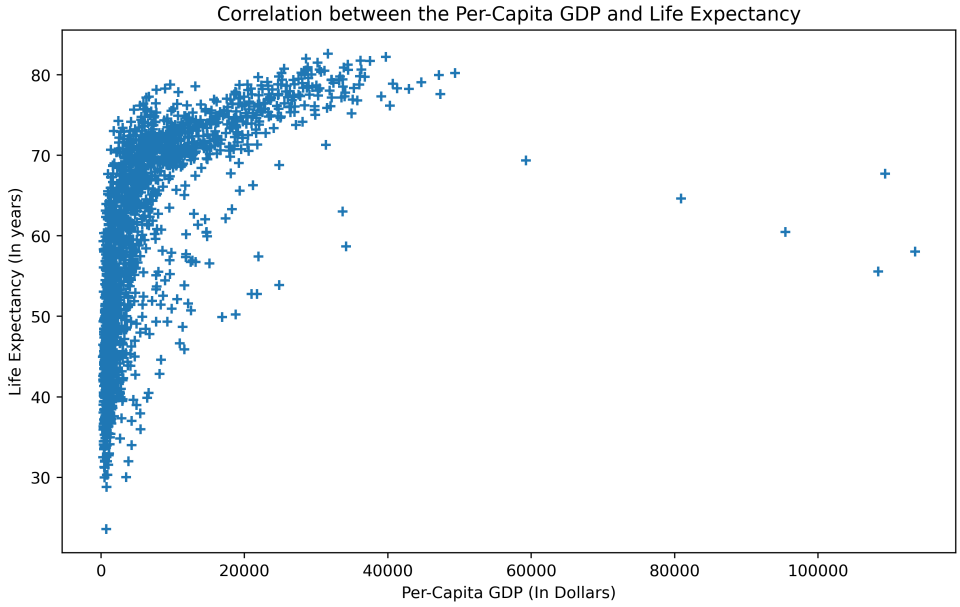


Figure 6: Correlation between the Per-Capita GDP and Life Expectancy

8. Heatmaps

   Seaborn is a Python visualization library based on Matplotlib and provides a high-level interface for drawing attractive statistical graphics. Here it can be used to build heatmaps that facilitate the plotting of three variables, one continuous and two factors in which a color scale is created to highlight the magnitude of the continuous variable.

   First, we will consider the average per capita GDP for each continent over time to create a heatmap.

```
per_continent_mean_gdp_df = gapminder_df.groupby(['continent', 'year'],
                            as_index=False)['gdpPercap'].mean()

plt.figure(figsize=(12, 8))
heatmap = per_continent_mean_gdp_df.pivot(index="year",
                                          columns="continent",
                                          values="gdpPercap")
heatmap = heatmap.sort_index(ascending=False)
heatmap_graph = sns.heatmap(heatmap, annot=True, fmt=".2f", cmap='YlGnBu',
                            cbar_kws={'label': "gdpPercap"})

plt.title("Average Per-Capita GDP for Each Continent over Time");
```

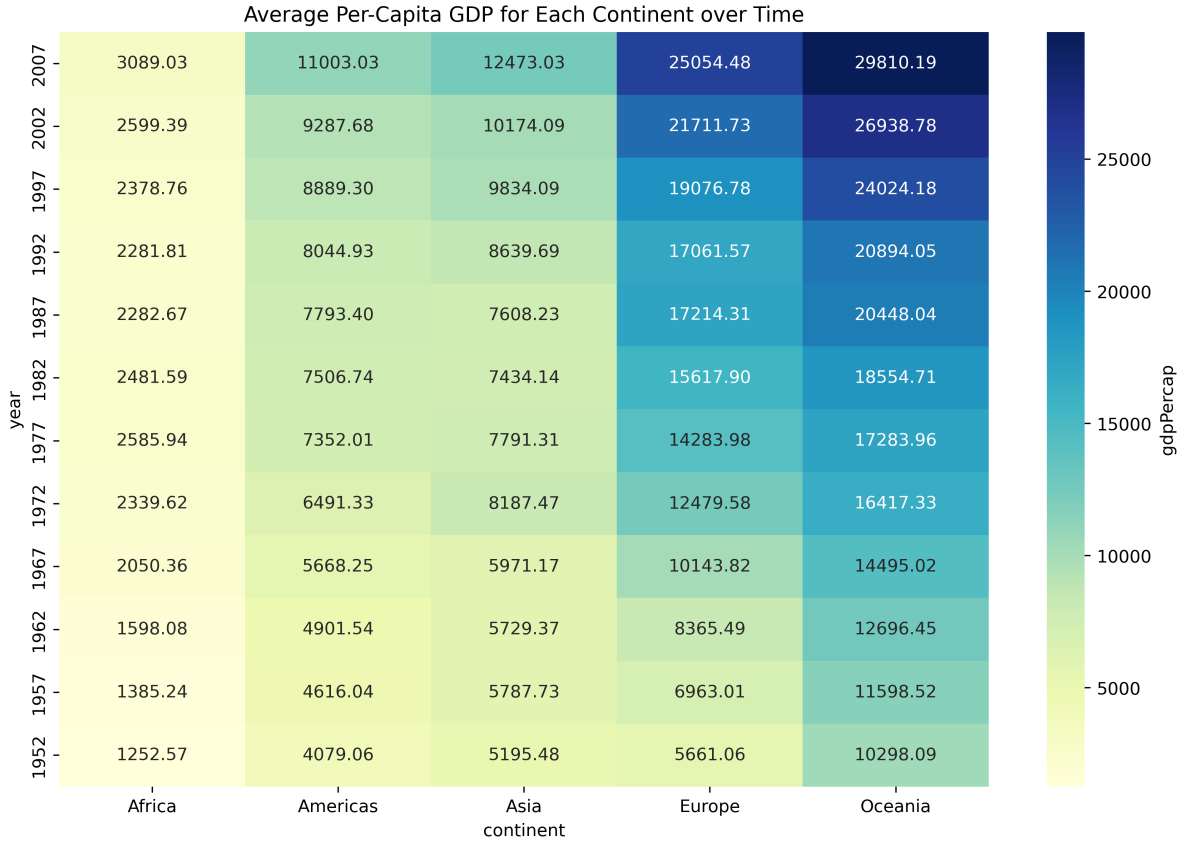


Figure 7: Heatmap for average Per-Capita GDP for each continent over time

9. Regression plots

   Furthermore, Seaborn includes many useful built-in plots such as regression plots. In this case, all data values are plotted using a scatter plot, and a regression line is fit and plotted on that data.

```
sns.regplot(x='year', y='lifeExp', data=gapminder_df, color='teal')
plt.title('Distribution of Life Expectancy over Year')
plt.xlabel('Year')
plt.ylabel('Life Expectancy');
```

10. Kernel Density Estimate plots

    A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analagous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.

    Let us plot a kernel density estimation plot and histogram (as shown in Figure 2) for per-capita GDP in 2007 side-by-side using subplots.

```
plt.figure(figsize=(15, 5))

plt.subplot(1, 2, 1)
plt.hist(recent_year_df['gdpPercap'])
plt.xlabel('Per-Capita GDP')
plt.ylabel("Number of Countries")
plt.title('Histogram for the Distribution of Global Per-Capita GDP in 2007')

plt.subplot(1, 2, 2)
sns.kdeplot(recent_year_df['gdpPercap'], shade=True)
```

```
plt.title('Kernel Density Estimate Plot for the Distribution of Global
           Per-Capita GDP in 2007');
```

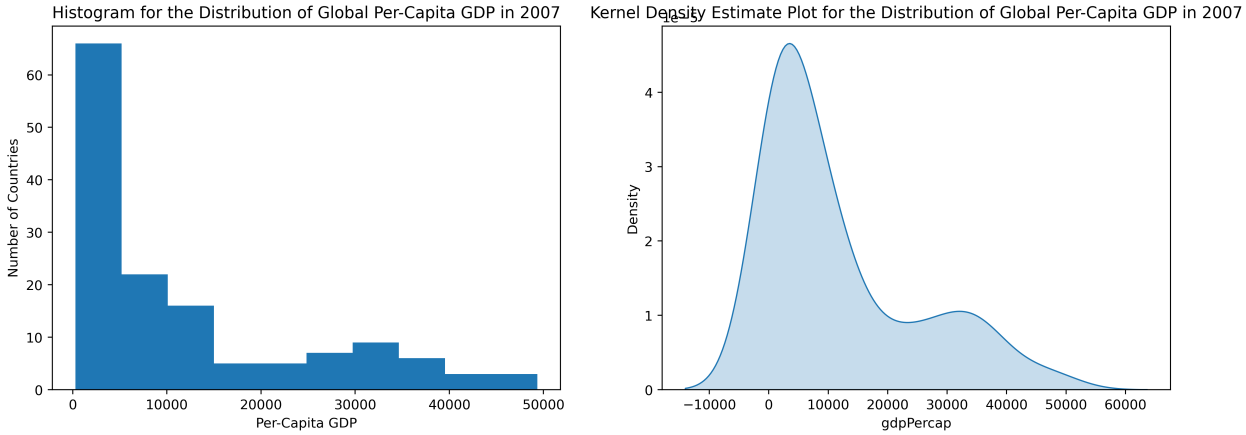


Figure 8: Distribution of Global Per-Capita GDP in 2007 using a histogram and KDE

11. Saving plots

    Matplotlib plots can be saved as image files using the following code snippet:

    ```
    plt.savefig('plot.png', dpi=400, bbox_inches ='tight')
    ```

---

**TODO 3:**

1. Load the winequality_red.csv file. (this is a dataset from UCI Machine Learning Repository open datasets)

   Hint: `df = pd.read_csv('winequality_red.csv', sep=';')`

2. Explore the correlation between each attribute and quality of the wine graphically. (take the direction and magnitude of correlation)

3. Visualize the relationship between each pair of attribute using a scatter matrix.

4. Visualize the distribution of data for each attribute using a box plot for each class.

   Use different colors to denote each class and your figures must contain titles, axis labels, and appropriate legends.

---

***** End of Labsheet *****