ELEC 292
Lab 4: Visualization

Goal:

This lab includes 2 main components. First, our goal is to visualize a widely used 'Heart Disease Dataset' to get some high-level information about the distribution of the data. Next, we aim to apply PCA and t-SNE on another widely used dataset called the 'wine quality dataset', with the foal of performing dimensionality reduction.

Instructions:

This lab includes 10 questions. In questions 1, 3, 4, 5, 6, 8, and 9 you need to write code in Python and generate the desired outputs. However, in questions 2, 7, and 10, you only have to answer the specified questions.

Deliverables:

1. A report which contains:
    a. Screen shots of your code and the outputs for questions 1, 3, 4, 5, 6, 8, and 9
    b. Your answers to questions 2, 7, and 10.
2. Your executable code in the .py format.

***** Please follow the following steps carefully – failure to do so will result in your lab not being graded:**

- On OnQ, go to **Communications**
- Then go to **Groups**
- Click on **View Available Groups**
- Find the **Lab 4** heading
- Under Lab 4, **ALL members of your team** must join a group (**the same group number**). Note that if someone does not join the group, they will not receive the grade for the lab
- Now, when you go to **Assessments > Assignments**, next to Lab 4, you should see your group number
- This will allow for one set of submissions to be used for everyone in the same group, and the grade and comments to be shared across all members

Libraries

In this lab, we need the following Python libraries:

1. Matplotlib: The famous Python library for data visualization.

2. Pandas: A widely used library for manipulating numerical tables. This library is built on top of NumPy and Matplotlib libraries.

To install these libraries, use the previous ELEC390 environment which you have already created in the previous labs. Open the terminal of your ELEC390 environment in Anaconda and type the following commands to install the libraries:

Matplotlib:

```
conda install -c conda-forge matplotlib
```

Pandas:

```
conda install -c anaconda pandas
```

Like previous labs, please check the environment and make sure that you have installed the mentioned libraries correctly.

Part 1: Gaining High-level Insights from the Heart Disease Dataset

In this part of the lab, we aim to create histograms, smooth histograms (also known as density plots), box plots, and scatter matrix plots of the Heart Disease Dataset. These plots help us gain a better high-level understanding of the dataset, which can potentially help us down the road when processing the dataset or using it to build a product.

Histogram Plots

Pandas is built on Matplotlib. You can therefore directly use Pandas for visualization purposes. Here, we aim to create the histogram plots using both Pandas and Matplotlib. However, for the other plots, we will only use Pandas.

Question 1

Follow the below steps to create a histogram plot of the Heart Disease Dataset. Don't forget to put '# Question 1' at the beginning of your code.

Step 1. Download the Heart Disease Dataset from the following address and move it to the same folder that you've chosen to create the new project in PyCharm (current directory):

https://drive.google.com/file/d/1S6P5HiD2UeO2Xj6g8_BxClK_KtEWsc6h/view?usp=sharing

Step 2. Import matplotlib.pyplot as plt. Also, import pandas.

Step 3. Type:
dataset = pd.read_csv(?)
and substitute '?' with the proper statement to read the dataset.

Step 4. Use the following code to separate the labels from the data:

```
data = dataset.iloc[?, ?]
labels = dataset.iloc[?, ?]
```

Step 5. Use the following code to create a plot with a 4 by 4 grid, and set the figure size to 20 by 10.

```
fig, ax = plt.subplots(ncols=?, nrows=?, figsize=?)
```

Please note that in the code above, 'ax' object will be a 4 by 4 matrix, which can be indexed.

Step 6. In the previous step, we created 'ax' and 'fig' objects. Now, everything is ready to create the histogram plots inside these objects. To this end, we should apply the '.hist' method to the data:

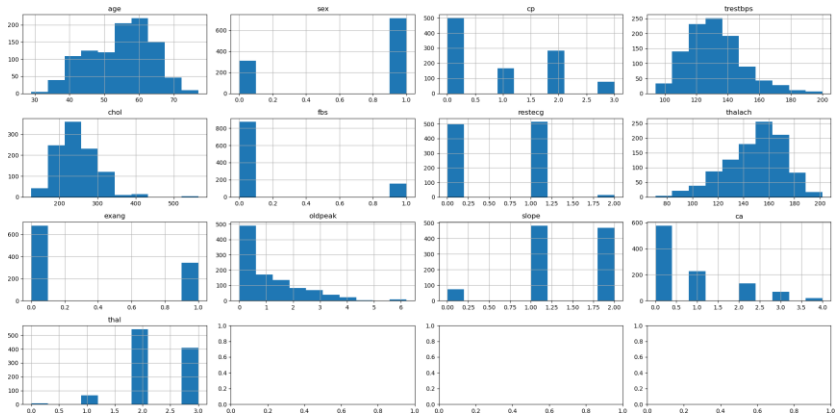
```
data.hist(ax=ax.flatten()[?:?])
```

To complete the code above, since there are 13 features inside the dataset (data has 13 columns), we should pass 13 'ax' objects out of 16 to the '.hist' method. Hence, index 'ax.flatten()[?:?]' such that it only includes 13 ax objects.

Step 7. Use the following code to make the layout of the plot tight and to render the plot:

```
fig.tight_layout()
plt.show()
```

The result should look like this:



Question2

Based on the histograms you just created, answer the following questions:

A. Are the majority of patients older than 40, or younger?
B. If we select one patient from this dataset randomly, what is their age the highest likelihood? (this question may have more than 1 valid answer – providing only one possible answer is enough)
C. What can you infer from the 'chol' (serum cholesterol) histogram about the cholesterol distribution of the patients?
D. If a feature has only two levels, we can call it a binary feature. What are the binary features in the Heart Disease Dataset?

Question 3

Follow the steps below to create histogram plots for the Heart Disease Dataset. Don't forget to put '# Question 3' before your code.

Step 1. For creating histogram plots using Matplotlib, first, follow same steps 1 to 5 of the previous part (Question 1). That is, you should read the dataset, separate the data from the labels, and create 4 by 4 'fig' and 'ax' objects:

```
dataset = pd.read_csv(?)
data = dataset.iloc[?:?]
labels = dataset.iloc[?, ?]
fig, ax = plt.subplots(ncols=?, nrows=?, figsize=?)
```

Step 2. Matplotlib has a '.hist()' method, which creates a histogram plot. However, it accepts only 1 'ax' object at a time. Therefore, since we need to create 13 histogram plots, we need to use a 'for' loop which iterates from 0 to 12. Complete the following code, and replace '?' with the right terms:

```
for i in range(?, ?):
```

Step 3. Inside the for loop, we need to create histogram plots for each column of our dataset. Complete the following code (remember this code must be written inside the for loop and **indented**):

```
    ax.flatten()[i].hist(data.iloc[?, ?])
```

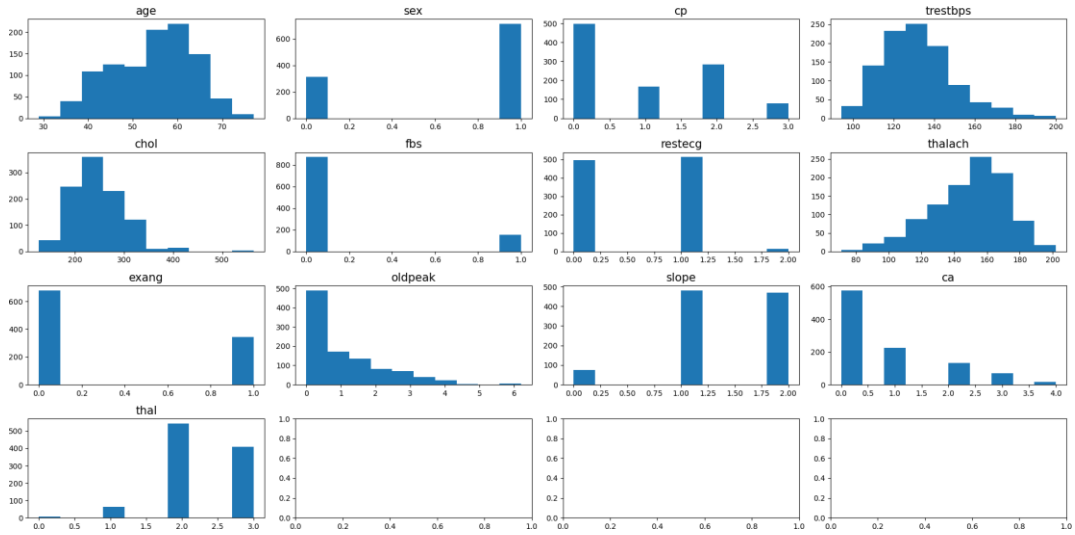
Step 4. Inside the 'for' loop, we also need to create the titles for each histogram. To this end, we can use the column names of data (remember 'data' variable is a data frame). Complete the following code (remember this code must be written inside the for loop):

```
    ax.flatten()[i].set_title(data.columns[?], fontsize=15)
```

Step 5. We are now done with the ‘for’ loop. Use the following code to make the layout of the plot tight and to render it:

```
fig.tight_layout()
plt.show()
```

The result should look like this:



Continuous Histogram Plots

In this part, we aim to create continuous histogram plots for the Heart Disease Dataset.

Question 4. Follow the steps below to create the continuous histogram plots. Don’t forget to put ‘# Question 4’ before your code.

Step 1. Like the previous sections, complete the following to read the dataset, separate the data from the labels, and create a 4 by 4 ‘fig’ and ‘ax’ objects:

```
dataset = pd.read_csv(?)
data = dataset.iloc[?:?]
labels = dataset.iloc[?, ?]
fig, ax = plt.subplots(ncols=?, nrows=?, figsize=?)
```

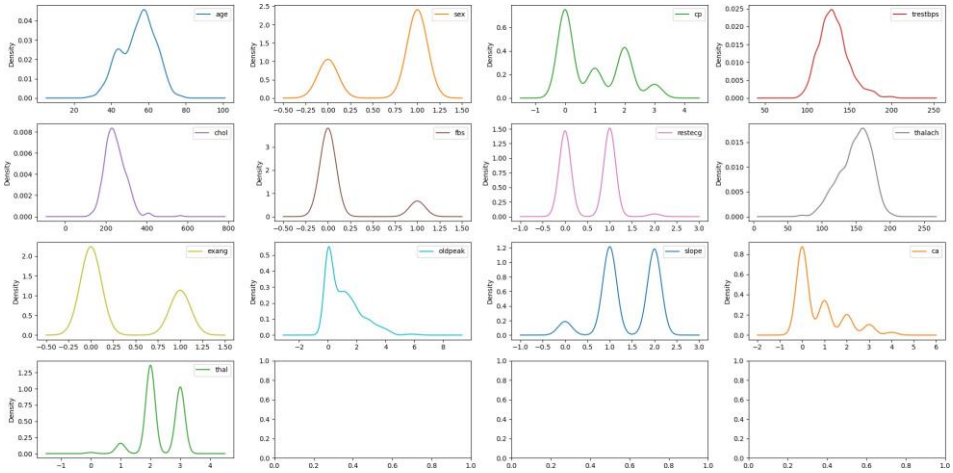
Step 2. We need to use the ‘.plot’ method on ‘data’ to create continues histogram plots. To do so, you should pass 13 axes objects (equal to the number of the columns in ‘data’) to the ‘.plot’ method. To do so, please complete the following code:

```
data.plot(ax=ax.flatten()[?:?], kind='density', subplots=True, sharex=False)
```

Step 3. Use the following code to make the layout of the plot tight and to render the plot:

```
fig.tight_layout()
plt.show()
```

The result should look like this:



Box Plots

In this part of the lab, we aim to create the box plots for the Heart Disease Dataset.

Question 5. Follow the steps below to create the box plots. Don't forget to put '# Question 5' before your code.

Step 1. Like the previous sections, complete the following to read the dataset, separate the data from the labels, and create a 4 by 4 'fig' and 'ax' objects:

```
dataset = pd.read_csv(?)
data = dataset.iloc[?:?]
labels = dataset.iloc[?, ?]
fig, ax = plt.subplots(ncols=?, nrows=?, figsize=?)
```

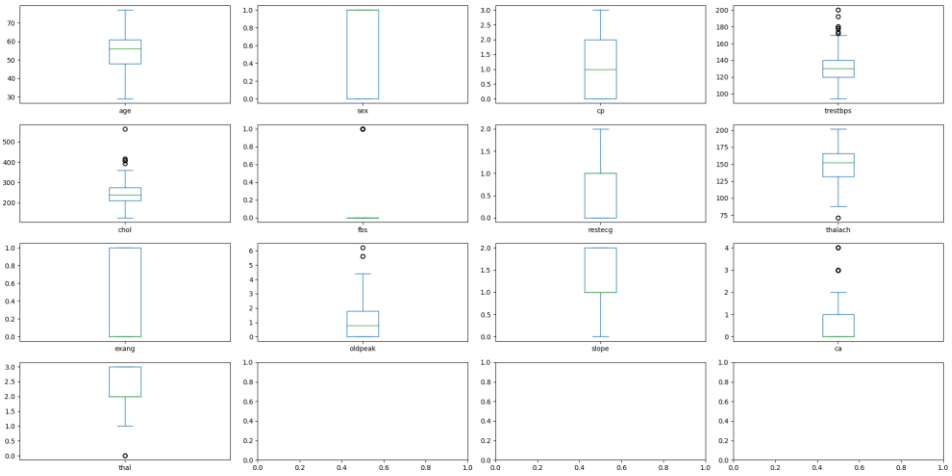
Step 2. We need to use the '.plot' method on the 'data' to create the box plots. To do so, you should pass 13 axes objects (equal to the number of the columns in 'data') to the '.plot' method. Complete the following code:

```
data.plot(ax=ax.flatten()[?:?], kind='box', subplots=True, sharex=False, sharey=False)
```

Step 3. Please use the following code to make the layout of the plot tight and to render the plot:

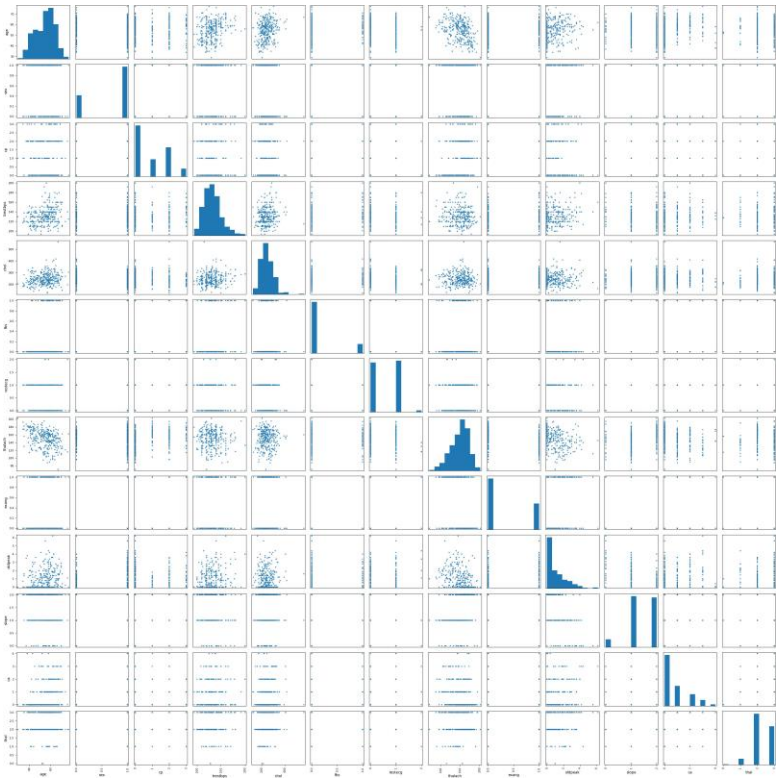
```
fig.tight_layout()
plt.show()
```

The result should look like this:



Scatter Matrix Plots

Scatter matrix plots are used to investigate the relationships between the features within a dataset. In other words, it shows how one feature is affected by another. The following is the scatter matrix plot of the Heart Disease Dataset:



As you can see, to create such a plot, we need 13*13=169 Axes objects.

Question 6

Follow the steps below to create the scatter matrix plots for the Heart Disease Dataset. Don't forget to put '# Question 6' before your code.

Step 1. Like the previous sections, complete the following to read the dataset, separate the data from the labels, and create **13 by 13** 'fig' and 'ax' objects. Also, this time, set the figsize to 30 by 30:

```
dataset = pd.read_csv(?)
data = dataset.iloc[?:?]
labels = dataset.iloc[?, ?]
fig, ax = plt.subplots(ncols=?, nrows=?, figsize=?)
```

Step 2. We need to use 'pd.plotting.scatter_matrix()' function to create the scatter matrix plot. To do so, we need to pass all 169 axes objects to the 'pd.plotting.scatter_matrix()' method. Please complete the following code:

```
pd.plotting.scatter_matrix(data,  ax=?)
```

Step 3. Use the following code to make the layout of the plot tight and to render the plot:

```
fig.tight_layout()
plt.show()
```

Question7

Based on the scatter matrix plot you just obtained, answer the following questions.

A. If by increasing one variable, the other variable also increases, we can infer that they have a *positive correlation*. If by increasing one, the other decreases, they have negative correlation. What kind of correlation does thalach (maximum heart rate) have with age?
B. Do thalach and chol have a strong correlation (either positive or negative)? Explain your answer.

Part 2: PCA and t-SNE for wine quality dataset

Question 8

Download the Wine Quality Dataset from the following address:

https://drive.google.com/file/d/1yRFAuDf0HKXo2xzI-P_a_zkZwIXgf6Ks/view?usp=sharing

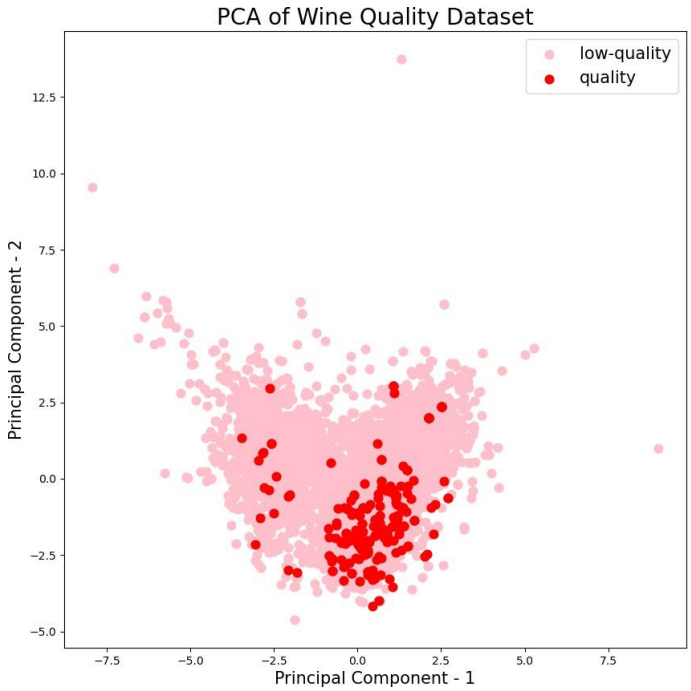
Now, complete the following steps:

Step 1. Read the dataset as a data frame.

Step 2. As the first column of the dataset contains only strings (‘red’ or ‘white’) and is not useful for our purpose, drop the entire first column from the data frame.

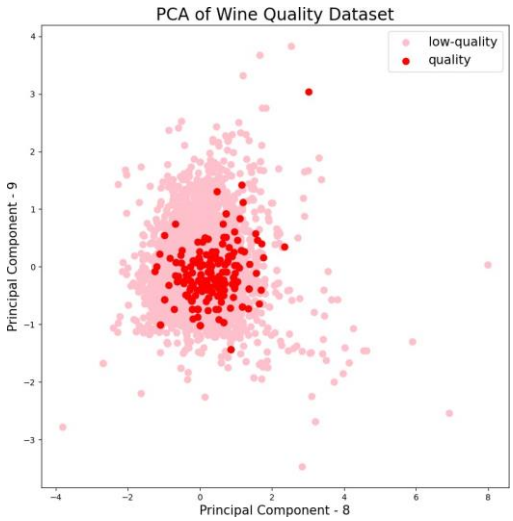
Step 3. The column that contains the target for this dataset is named ‘quality’. The wine *quality* that is contained in this column is a number between 0 and 10 (higher scores indicate better quality). Suppose that we define *high-quality wine* with a score equal to or above 8, and *low-quality wine* with a score of 7 or less. Based on this definition, in the ‘quality’ column, change the values for all ‘high-quality’ wines to 1 and all ‘low-quality’ wines to 0.

Step 4. Now, based on what you learned in lectures, apply PCA and t-SNE on the dataset. Set ‘n_components’ to **2 in both t-SNE and PCA. For t-SNE, set perplexity to 30. Plot the resulting figures.** In your plot, set the colors for low-quality wines to pink and high-quality wine to red. Your results for PCA should look like the following:



Question 9

In the previous question, we applied PCA to the wine quality dataset with ‘n_component=2’. In other words, we plotted the first two *principal components* (new dimensions) of the resulting PCA (*x* axis shows principal component 1 and the *y* axis represents the principal component 2). However, for this dataset, we have *11* principal components (as it has 11 columns after dropping the first column). **Revise your previous code to plot the principal component 8 vs. principal component 9.** Therefore, in the resulting figure, the *x* axis must represent principal component 8 and the *y* axis must show principal component 9. The result will look like this:



Question 10

In Question 7, we plotted principal components (new dimensions) 1 vs. 2, while in Question 8, we plotted principal components 8 vs. 9. Which of these two scenarios carries more information from the wine quality dataset? Why?