1 Python Lab Exercise #2

1.1 Objectives:

- Load .csv files into `pandas` DataFrames
- Describe and manipulate data in Series and DataFrames
- Visualize data using DataFrame methods and `matplotlib`

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1.2 What is Pandas?

Pandas, as [the Anaconda docs (https://docs.anaconda.com/anaconda/packages/py3.7_osx-64/)](https://docs.anaconda.com/anaconda/packages/py3.7_osx-64/) tell us, offers us "High-performance, easy-to-use data structures and data analysis tools." It's something like "Excel for Python", but it's quite a bit more powerful.

Let's read in the heart dataset.

Pandas has many methods for reading different types of files. Note that here we have a .csv file.

Read about this dataset [here (https://www.kaggle.com/ronitf/heart-disease-uci)](https://www.kaggle.com/ronitf/heart-disease-uci).

In [ ]:
```
heart_df = pd.read_csv('heart.csv')
```

The output of the `.read_csv()` function is a pandas *DataFrame*, which has a familiar tabaular structure of rows and columns.

In [ ]:
```
type(heart_df)
```

In [ ]:
```
heart_df
```

1.3 DataFrames and Series

Two main types of pandas objects are the DataFrame and the Series, the latter being in effect a single column of the former:

In [ ]:
```
age_series = heart_df['age']
type(age_series)
```

Notice how we can isolate a column of our DataFrame simply by using square brackets together with the name of the column.

Both Series and DataFrames have an *index* as well:

In [ ]:
```
heart_df.index
```

In [ ]:
```
age_series.index
```

Pandas is built on top of NumPy, and we can always access the NumPy array underlying a DataFrame using `.values` .

In [ ]:
```
heart_df.values
```

1.4 Basic DataFrame Attributes and Methods

1.4.1 `.head()`

In [ ]:
```
heart_df.head()
```

1.4.2 `.tail()`

In [ ]:
```
heart_df.tail()
```

1.4.3 `.info()`

In [ ]:
```
heart_df.info()
```

1.4.4 .describe()

In [ ]:
```
heart_df.describe()
```

1.4.5 .dtypes

In [ ]:
```
heart_df.dtypes
```

1.4.6 .shape

In [ ]:
```
heart_df.shape
```

1.4.7 Exploratory Plots

Let's make ourselves a histogram of ages:

In [ ]:
```
sns.set_style('darkgrid')
sns.distplot(a=heart_df['age']);
# For more recent versions of seaborn:
# sns.histplot(data=heart_df['age'], kde=True);
```

And while we're at it let's do a scatter plot of maximum heart rate vs. age:

In [ ]:
```
sns.scatterplot(x=heart_df['age'], y=heart_df['thalach']);
```

1.5 Adding to a DataFrame

1.5.1 Adding Rows

Here are two rows that our engineer accidentally left out of the .csv file, expressed as a Python dictionary:

In [ ]:
```
extra_rows = {'age': [40, 30], 'sex': [1, 0], 'cp': [0, 0], 'trestbps': [120, 130
              'chol': [240, 200],
             'fbs': [0, 0], 'restecg': [1, 0], 'thalach': [120, 122], 'exang': [0
              'oldpeak': [0.1, 1.0], 'slope': [1, 1], 'ca': [0, 1], 'thal': [2, 3
              'target': [0, 0]}
extra_rows
```

How can we add this to the bottom of our dataset?

In [ ]:
```
# Let's first turn this into a DataFrame.
# We can use the .from_dict() method.

missing = pd.DataFrame(extra_rows)
missing
```

In [ ]:
```
# Now we just need to concatenate the two DataFrames together.
# Note the `ignore_index` parameter! We'll set that to True.

heart_augmented = pd.concat([heart_df, missing],
                            ignore_index=True)
```

In [ ]:
```
# Let's check the end to make sure we were successful!

heart_augmented.tail()
```

1.5.2 Adding Columns

Adding a column is very easy in `pandas`. Let's add a new column to our dataset called "test", and set all of its values to 0.

In [ ]:
```
heart_augmented['test'] = 0
```

In [ ]:
```
heart_augmented.head()
```

I can also add columns whose values are functions of existing columns.

Suppose I want to add the cholesterol column ("chol") to the resting systolic blood pressure column ("trestbps"):

In [ ]:
```
heart_augmented['chol+trestbps'] = heart_augmented['chol'] + heart_augmented['tre
```

In [ ]:
```
heart_augmented.head()
```

1.6 Filtering

We can use filtering techniques to see only certain rows of our data. If we wanted to see only the rows for patients 70 years of age or older, we can simply type:

In [ ]:
```
heart_augmented['age'] >= 70
```

In [ ]:
```
heart_augmented[heart_augmented['age'] >= 70]
```

Use '&' for "and" and '|' for "or".

1.6.1 Exercise

Display the patients who are 70 or over as well as the patients whose trestbps score is greater than 170.

In [ ]:
```
# Enter your code here
```

1.6.2 Exploratory Plot

Using the subframe we just made, let's make a scatter plot of their cholesterol levels vs. age and color by sex:

In [ ]:
```
at_risk = #[ANSWER FROM EXERCISE]

sns.scatterplot(data=at_risk, x='age', y='chol', hue='sex');
```

1.6.3 `.loc` and `.iloc`

We can use `.loc` to get, say, the first ten values of the age and resting blood pressure ("trestbps") columns:

In [ ]:
```
heart_augmented.loc
```

In [ ]:
```
heart_augmented.loc[:9, ['age', 'trestbps']]
```

`.iloc` is used for selecting locations in the DataFrame **by number**:

In [ ]:
```
heart_augmented.iloc
```

In [ ]:
```
heart_augmented.iloc[3, 0]
```

In [ ]:
```
heart_augmented.head()
```

1.6.4 Exercise

How would we get the same slice as just above by using .iloc() instead of .loc()?

In [ ]:
```
# Enter your code here
```

1.7 Statistics

1.7.1 .mean()

In [ ]:
```
heart_augmented.mean()
```

Be careful! Some of these will are not straightforwardly interpretable. What does an average "sex" of 0.682 mean?

1.7.2 .min()

In [ ]:
```
heart_augmented.min()
```

1.7.3 .max()

In [ ]:
```
heart_augmented.max()
```

1.8 Series Methods

1.8.1 .value_counts()

How many different values does slope have? What about sex? And target?

In [ ]:
```
heart_augmented['slope'].value_counts()
```

In [ ]:
```
heart_augmented['sex'].value_counts()
```

1.8.2 .sort_values()

In [ ]:
```
heart_augmented['age'].sort_values()
```

1.9 pandas -Native Plotting

The `.plot()` and `.hist()` methods available for DataFrames use a wrapper around `matplotlib`:

In [ ]:
```
heart_augmented.plot(x='age', y='trestbps', kind='scatter');
```

In [ ]:
```
heart_augmented.hist(column='chol');
```

1.9.1 Exercises

1. Make a bar plot of "age" vs. "slope" for the `heart_augmented` DataFrame.

In [ ]:
```
# Enter your code here
```

2. Make a histogram of ages for **just the men** in `heart_augmented` (heart_augmented['sex']=1).

In [ ]:
```
# Enter your code here
```

3. Make separate scatter plots of cholesterol vs. resting systolic blood pressure for the target=0 and the target=1 groups. Put both plots on the same figure and give each an appropriate title.

In [ ]:
```
# Enter your code here
```