FIT1043 Introduction to Data Science

Week 2 Lectures

Python for Data Science

Specific libraries that are considered as the “starter pack” for Data Science:

- **Numpy**: Scientific computing, support for multi-dimensional arrays
- **Pandas**: Data structures as well as operations for manipulating numerical tables.
- **Matplotlib**: library for visualization
- **Scikit-learn**: Python machine learning library that provides the tools for data mining and data analysis

For some, you may also want to look at

- **NLTK**: Natural Language ToolKit to work with human language data

Loading Libraries

The general syntax to include a library:

In [ ]:

```
import numpy as np
import pandas as pd
# The next two are equivalent
from matplotlib import pyplot as plt
import matplotlib.pyplot as plt
```

Reading and Writing Data

Data Science needs DATA

- Reading data
- Writing data

We can read data from different sources

- Flat files
- CSV files
- Excel files
- Image files
- Relational databases
- NoSQL databases
- Web

Reading from CSV

Python has a built in CSV reader but for Data Science purposes, we will use the pandas library.

- Assuming your file name is `filename.csv` (it should be on your Moodle)

In [ ]:

```
import pandas as pd
# You need to download the file filename.csv to your Jupyter Notebook start folder.
data = pd.read_csv("filename.csv")
data.head()
```

In [ ]:

```
X = data[["Age"]]
print(X)
```

Usual first step when we first obtain any data is to get a description or a summary of it.

Sometimes, referred to as **five number summary** if the data is numeric.

- minimum
- maximum
- median
- 1st quartile
- 3rd quartile

Let's work with `pandas` DataFrames.

In [ ]:

```
# data was already read from above
df = pd.DataFrame(data)
print(df)
```

In [ ]:

```
df.describe()
```

Working with DataFrames (Basic)

Select a column by using its column name:

In [ ]:

```
df['Name']
```

Select multiple columns using an list of column names:

In [ ]:

```
df[['Name', 'Survived']]
```

Select a value using the column name and row index:

In [ ]:

```
df['Name'][3]
```

Select a particular row from the table:

In [ ]:

```
df.loc[2]
```

Select all rows with a particular value in one of the columns:

In [ ]:

```
df.loc[df['Age'] <= 6]
```

Writing (Saving) the Data

Assuming you just want to analyse a part of the data and you want to save a resulting data frame to a CSV file.

In [ ]:

```
df2 = df.loc[df['Age'] >= 12]
# Use your own folder location, it will definitely differ from mine unless you are also
Ian Tan :)
df2.to_csv (r'C:\Users\Ian Tan\Desktop\output.csv', index = None, header=True)
```

Go and check on your folder if the file is save correctly. Note that you can open the CSV file using a text editor (in Windows, you can use Notepad).

We have now read, describe, basic data exploration and save the data.

Pause Take sometime to review what you have learnt so far

Working with Data

There are some basic data pre-processing that are usually done or at least taken into consideration.

- Removing duplicates
- Categorical data
- Dealing with dates
- Missing data
- Subsetting data
- Concatenating
- Transforming
- Aggregating

More will be explored in week 4

Categorical Data

A categorical data is one that has a specific value from a limited set of values. The options are fixed.

- A ticket class is generally categorical, i.e. 1st class, 2nd class & 3rd class.

In [ ]:

```
df.loc[df['Pclass'] == 1]
```

We can create our own categories, e.g.

In [ ]:

```
import pandas as pd
tix_class = pd.Series(['1st','2nd','3rd'], dtype='category')
```

Subsetting Data

Extract only those that survived

In [ ]:

```
df.loc[df['Survived'] == 1]
```

What does the code below return? Before you run it, discuss and determine it.

In [ ]:

```
df.loc[(df['Sex'] == 'female') & (df['Survived'] == 1)]
```

Slicing Data

Slice rows by row index.

In [ ]:

```
df[:5]
```

In [ ]:

```
df[3:10]
```

If we only want certain columns, e.g. Age, Name, Sex, Survived

In [ ]:

```
df.loc[:, ('Age','Name','Sex','Survived')]
```

Aggregating

Like our 5 number statistic, we can also obtain aggregated values for columns. The total fare can be easily obtained by

In [ ]:

```
df['Fare'].sum()
```

Or we can get the average age of the passenger by

In [ ]:

```
df['Age'].mean()
```

Check the answers against the `df.describe()` earlier

Like in SQL, we often want to know the aggregated values for certain values from another column. Similarly, we can use the `groupby()` function.:

In [ ]:

```
df.groupby('Sex')['Age'].mean()
```

What does the following mean?

In [ ]:

```
df.loc[df['Survived']==1].groupby('Sex')['Age'].mean()
```

Compare it with the previous statement, what can you tell from it?

Of the females that survived, most are elderly while for the males, the survivors are younger.

We will be using Python for the next few weeks

- MatPlotLib
- Scikit-Learn

You can easily look for Python resources online, to be specific, Python for Data Science. An excellent online course will be from DataCamp