Data Mining and Decision Systems
Workshop 4
Seaborn Visualisation

Aims of the workshop

Introduced in last week's lecture, Seaborn will be the basis for today's workshop. The aim of this session is to get to grips with this library using some of the coding and functions demonstrated in the live coding session.

We will introduce a means for loading standard datasets provided by Seaborn, such as the restaurant tip dataset. This may be useful if you wish to practice on data not seen in workshops thus far.

Please see 'Useful Information' below on how to lookup certain Python functionality. The concept behind this workshop is about discovery, and experimentation surrounding topics covered so far.

Feel free to discuss the work with peers, or with any member of the teaching staff.

Useful Information

Throughout this workshop you may find the following useful.

Python Documentation

https://docs.python.org/3.7/

This allows you to lookup core language features of Python 3.7 as well as tangential information about the Python Language.

Pandas Documentation

https://pandas.pydata.org/pandas-docs/stable/index.html

DataFrames:
https://pandas.pydata.org/pandas-docs/stable/reference/frame.html

Series:
https://pandas.pydata.org/pandas-docs/stable/reference/series.html

Similarly to the above, the pandas library itself is well-documented outlining function signatures, alongside example use-cases.

Functions of note: read_csv, [ ], [[ ]], copy, distplot, catplot, boxplot, set_xscale, set_yscale, load_dataset, corr, heatmap, scatterplot, pairplot,

Useful indexing introduction:
https://www.geeksforgeeks.org/indexing-and-selecting-data-with-pandas/

Seaborn

Seaborn is a high-level plotting library for Python. Written on top of Matplotlib it enables rich and intricate plots whilst maintaining readability and ease of use. You can access the library's web page at the following link:
https://seaborn.pydata.org/

For API Reference, consider: https://seaborn.pydata.org/api.html

You may find https://seaborn.pydata.org/tutorial.html useful for understanding more functionality, and introducing you to each aspect.

Matplotlib

Matplotlib is a powerful 2D plotting library for Python. It is the de-facto standard for the community. Whilst matplotlib is considered powerful, this comes at the expense of verbosity. Those wishing to further manipulate their diagrams and plots may wish to seek out matplotlib documentation (https://matplotlib.org/3.1.1/api/index.html) to enable more advanced graphics manipulation.

Reminder

We encourage you to discuss the contents of the workshop with the delivery team, and any findings you gather from the session.

Workshops are not isolated, if you have questions from previous weeks, or lecture content, please come and talk to us.

The contents of this workshop are not intended to be 100% complete within the 2 hours; as such it’s expected that some of this work be completed outside of the session. Exercises herein represent an example of what to do; feel free to expand upon this.

Running A Jupyter/IPython Notebook

A jupyter notebook is a server which runs on the local machine. It will use whichever directory it is invoked within as the root folder. It is then able to see all sub-folders within that. This is currently installed on the lab machines; however, if you are wishing to install this at home, you may need to follow pip instructions for installing jupyter.

Jupyter Notebook can be launched from the command line by invoking:

```
jupyter notebook
```

This will start a server, typically on http://localhost:8888/tree, and should automatically open a new tab in the browser.

Jupyter notebook server uses a token, generated at launch, to authenticate access. By default, the notebook server is accessible to anybody with network access to your machine over the port, with the correct token.

If you are asked for a password or token, you can always invoke:

```
jupyter notebook list
```

This will provide you a list of currently active Notebook servers (you can run multiple, on different ports, for different folders), along with their token, all within a single URL.

NOTE: You may wish to `cd` into an appropriate starting folder, such as a USB Stick, or a folder where you can store all of your Data Mining Workshop documents.

For each of the exercises below, please use a separate code cell. This allows you to compute just the given exercise and execute it, checking the output. Subsequent cells all share the same variable space - variables you assign in cell 1 will be available afterwards in other cells.

As with all workshops, after you read the original data in, create a copy to work with!

Exercise 1: Install the *seaborn* library using pip from a command line (or jupyter notebook cell). You may need to refer back to previous workshops.

Exercise 2 (Histograms): Load the titanic dataset. Plot a histogram (univariate) for each appropriate attribute within the dataset using distplot. Think back to previous workshops where you considered whether an attribute was categorical, or numeric; whether it has a large range of values (candidates for discretisation).

1. List the attributes you will be considering to plot for histograms. Why are these chosen? Are there any which could be categorical instead of numeric, or vice versa?
2. Perform this on the raw data. Does this cause any errors? If so, what are these? Are the plots accurate, or do they indicate some issues with the data? (Think back to large values of 0 for age from lecture).
3. Perform this on the cleaned data from previous workshop tasks. Is there a difference in the graphic? If so, what is this?
4. Change the bins from the default amount to higher and lower values. Remember you can pass a keyword argument to the function. How does this change the displayed plot? Do any new patterns emerge, or do any seem to disappear?

Exercise 3 (Categorical plots): For each categorical variable, use catplot

1. List the categorical variables you are considering, and why they are chosen.
2. Plot the bar chart (catplot) using **kind='count'** as the keyword argument. Are there any interesting comparisons?

Exercise 4 (Categorical vs Numeric): Using a boxplot look at variables which are categorical (x-axis) and numerical (y-axis).

You may find ax = sns.boxplot(...) useful. This provides a variable for the axes of the graph. From this we can set the x-axis and y-axis scales. E.g ax.set_xscale('log'). This may be useful as a logarithmic scale may improve readability.

1. Identify various combinations of attributes which fit this. E.g Embarked vs Fare.
2. What does this plot tell us about the Fare given where a passenger Embarked?
3. Plot other categorical attributes and numeric attribute combinations.

Exercise 5 (Scatter plot / Scatterplot matrix): Using seaborn’s *load_dataset* function, we can load some standard datasets which Seaborn provides.

tips = sns.load_dataset('tips')

1. Using the corr function generate a correlation table for this tips dataset.
2. Using a seaborn *heatmap* plot this correlation table.
3. Identify combinations of numeric attributes which seem to be correlated.
4. Confirm this by using scatterplot to plot the two attributes against each other. What is the correlation, is it positive or negative? Strong or Weak?
5. Look at other pairs of attributes from the heatmap, and consider their correlations.
6. Plot a scatterplot matrix (sns.pairplot) of the tips dataset. Some of the output seems strange, can you think of another plot type which might be more descriptive to show the relation between size and total_bill?

Exercise 6 (Hue): In the visualisation lecture we covered how you can add extra dimensionality, whilst maintaining the 2D spatial representation, by using colour for categories.

1. Consider the boxplots in Ex 4. What additional categorical variable could be useful to add to this?
2. Using the boxplot for Sex vs Fare, add Hue to the function call, specifying Pclass. Display the result. What can you see now about the data that wasn’t as obvious as before?
3. Plot the same variables again, except this time using x = 'Pclass', y='Fare', with Hue=’Sex’. This is the same data, just represented differently. Is the plot easier or harder to interpret than the one above? Can you see a clearer pattern now?

The Extended Exercises are optional, and are offered as an advanced supplement for those who have completed the existing work and wish to expand on their knowledge and challenge themselves further.

Extended Exercise 1: Look into matplotlib functionality, tweak the display of the plots to have larger xticks and yticks, and to have a bigger figsize.

Extended Exercise 2: Using the magic commands, use interactive mode to alter the display of the graphs from the exercises.

Extended Exercise 3: Look at seaborn documentation for 'pointplot'. This provides a means to estimate the mean of a given variable, and easily compare relationships and trends between groups of hue. Look at using this to represent a passenger's PClass against their survivorship; differentiated by their Sex.

What does this plot tell you about somebody's survivorship based on their Class and Sex. Is there a significant difference between classes? Is there a significant difference between Sex? Is this trend the same for all Classes? Could this extracted relationship be represented in a simpler way?