Fundamentals of Data Science. Workshop Number 1(Week-5) (24/Oct/2022).

Aims of the workshop.

Today, we started out our journey in data science by refreshing memories of the scientific method, having a look at a specific data science example (sports analytics) using limited data, and made a start at fitting lines to data sets (linear regression), along with looking at different types of data. In the weeks ahead, we will turn our attention to “data cleaning” more formally and make significant progress toward formal data management.

Workshop Timetable.

The workshops in Fundamentals of Data Science will be significantly different to other modules. We will use the time not only for coding, but also for discussion, small group activity, and tutorial activity. For this week, the timetable is given below.
Please note that this timetable should be taken as indicative only. We will modify the times according to how quickly things progress. Also note that we do not assign deadlines to these workshops as the activity is very open-ended.

You can use the live Teams channel for any Question/Help by using the link/ QR code below:
https://teams.microsoft.com/l/channel/19%3aRvDPf3g3nzxPmwoHz0jS_sYRbuSHGGnbp3UnCZTCKzQ1%40thread.tacv2/General?groupId=ff130d6f-1b76-41fc-afc8-b03afe435c28&tenantId=490a8197-7b83-4f10-89b9-83189be3835e

Time	Activity
15:00-15:05	Introductions.
15:05-15:20	Discussion / Tutorial on Sports Analytics Task.
15:20-18:00	Exercises.

Useful Information.

Throughout this workshop you may find the following useful.

Python Documentation

https://docs.python.org/3.8/

This allows you to lookup core language features of Python 3.8 as well as tangential information about the Python Language. We will refer you back to the Programming module by Brian Tompsett and Tareq Al Jaber for more notes on this.

Jupyter Notebook Basics

Jupyter itself offers some basic documentation for people new to the editor. These can be found on https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html

Jupyter can also use markdown cells for text input to describe things. If you wish to annotate each cell as to which Exercise it belongs to, you may find https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Working%20With%20Markdown%20Cells.html useful.

Reminder.

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

Note that workshops are not isolated undertakings!
If you have questions from previous weeks, or lecture content, please talk to us.

The contents of this workshop are not intended to be 100% complete within the session; 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.

15:00. Introductions.
The core teaching team for the Fundamentals of Data Science Workshops are:
Program leader: Dr. Tareq Al Jaber (t.al-jaber@hull.ac.uk)
Lecturers Dr. Tareq Al Jaber (t.al-jaber@hull.ac.uk), Dr Anna Zincenko (a.zincenko@hull.ac.uk) and Dr Aoife Curran (a.m.curran@hull.ac.uk)

GTAs Team: Samuel Rose; Ervands Mumdzjans; Victoria Sherratt; Laura Hunt; Lyra Hawkins; Ryan Alexander; Jevon Westcarr; Oliver Thompson; Georgia Lowes; Floyd Askin; Ifeoluwa Wuraola; Franky George; Lucy Smith; Elliot Howatson; Kuniko Azuma and Jack Cracknell.

15:05. Discussion (mini-tutorial / study group) on Sport Analytics Task.
In the week, we asked you to look at some data from the Australian Football League (AFL).

In this workshop, we'd like to discuss this exercise before starting on some coding tasks. The reason to do this is that we want to get students thinking more like data scientists in a given domain, and to provide some in-depth consolidation.

Open Questions:
(a) How would more data have helped you decide which team would have a better chance of winning?

(b) What sort of extra data would you have liked access to? And importantly: why?

(c) Is it possible to even derive an answer to the question posed with the data supplied? Why or why not?

(d) Further Contextualisation: If you owned a betting or gambling business, would you take bets on the game's outcome? If so, what would you look at and why? How would you compute what "odds" to give your customers?

15:20. Exercises.
For the coding tasks below, create a new notebook and name it something appropriate and/or memorable such as "DataScienceWorkshop1". I have tried to divide up these tasks in to "science" and "coding", although the differentiation between the two will get fuzzier as we progress.

Exercise 1.
It is fairly common to create some random data to test code out with. In science, this is often done so that people can test their code on fake input data to make sure it works before applying it to "real" data. The first exercise is to get to grips with generating some fake data. To do this, we are going to use the "random" package.

Coding.
Let's use the random package to generate 100 random integers between 1 and 100.
Below is how we'd generate 20 points – modify it to create 100.

```
import random
x1=[]
for i in range(20):
  x1.append(random.randint(1,100))
```

Exercise 2.
Coding. Look up how we would use "random" to create floating point random numbers, rather than integers. Generate 100 random numbers with one decimal place digit and store it in another list (e.g., in a list called "x2").

Exercise 3.
Coding. Let's generate 100 random coordinates. Use "random" to create 100 (x,y) coordinates, where x and y can range between 1 and 100, and they use one decimal place.

Exercise 4.
Science: If we tried to fit a line of best fit to the coordinates obtained in Exercise 3, what would you expect the result to be? Answer this either by yourself, in a small group, or talk to a GTAs about it before doing anything else.
Once you have answered this, justify your answer by attempting a linear regression on this data. Does it agree with your prediction? Why or why not?

Exercise 5.
Okay, so that line of best fit in Exercise 4 was hardly well fit at all. And that was to be expected. Let's see if we can cut down the data we've generated to provide a better chance for our code to fit the data.

Coding: Eliminate all data with y coordinate *outside* the range 40<y<60.
You might like to use an "if" statement to achieve this.
Store the data that falls within the range 40<y<60 into a new variable (e.g., y2).

Exercise 6.
Science. Will the new data – i.e. (x,y2) coordinates – have a better linear regression fit? Justify your answer.

Exercise 7.
Coding: Go ahead and make a linear regression to (x,y2). Is the result what you expected? Why or why not?

Exercise 8.
We're going to go a bit beyond linear regression now. We will produce some random data that should be fit by a parabola instead (i.e. an equation that looks like $y=ax^2+bx+c$, where a, b, and c are all constants). Try this:

```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(0)
X = 2 - 3 * np.random.normal(0, 1, 20)
Y = X - 2 * (X ** 2) + 0.5 * (X ** 3) + np.random.normal(-3, 3, 20)
plt.scatter(X,Y, s=10)
plt.show()
```

Hopefully you should see a set of random data points that look like a curve.

Exercise 9.
We will cover this process in more detail in lectures in the weeks ahead. However, for today's workshop, I'd like you to attempt to fit a parabola to the above data.
This can be achieved with POLYNOMIAL REGRESSION. We will investigate this in Week 9's lectures in more detail when we look at SKLEARN. For now, what I'd like you to do is slightly different approach (as with many programming tasks there is usually more than one way to do things, after all!).

#Firstly import the following if you haven't done so already:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import leastsq
```

#Now we have to define some function that will help us solve (and fit!) a parabola.

```
def func(params, x):
    a, b, c = params
    return a * x * x + b * x + c
```

Error function follows:

```
def error(params, x, y):
    return func(params, x) - y
```

And now the solution parameters:

```
def slovePara():
    p0 = [10, 10, 10]
    Para = leastsq(error, p0, args=(X, Y))
    return Para
```

#Finally the solution can be done as follows.

```
def solution():
    Para = slovePara()
    a, b, c = Para[0]
    print("a=",a," b=",b," c=",c)
    print("cost:" + str(Para[1]))
    print("The curve of solution is:")
    print("y="+str(round(a,2))+"x*x+"+str(round(b,2))+"x+"+str(c))
```

If you now call "solution()" you should see the solution to the parabola outputted.
If you've used the same random number seed as the example above, the solution will be:

```
a= -1.6285313195695865  b= 8.48492680502338  c= -6.119739882377941
```

Check to see if you can recover these values.
Extension Exercise: Add a curve on to a graph of your points with the above solution.

Exercise 10.
Coding.
How would you modify the above code to fit a third order polynomial?
(A third order polynomial would have the form: y = a*x^3 + b*x^x + c*x + d).

Exercise 11.
Science: how would you expect the fit to change as you added (or removed) more of those random points that were generated in Exercise 8 and fit in Exercise 9?

Exercise 12.
Science: How would you identify "outliers" prior to undertaking a fit?
Coding: Add in a clearly spurious point to Exercise 8. Re-do Exercise 9, but add to your code a way to "detect" a clear outlier. Feel free to discuss how you might do this on the chat in Teams, or with a GTAs.