Datasets and handling data in Python

Objective: The objective is to you acquainted with the exercise format of the course. Additionally, the aim is to familiarize yourself with the standard dataformat used in the course. Upon completing this exercise it is expected that you:

- Understand the format of the exercises and how the exercises are related to the reports.
- Can import data into Python and represent the data in course $\mathbf{X}$, $\mathbf{y}$ format.
- Can do common preprocessing steps for datasets.
- Have selected a proper dataset for use in the your project work (for the reports).

Material:

PYTHON Help: You can get help in your Python interpreter by typing `help(obj)` or you can explore source code by typing `source(obj)`, where `obj` is replaced with the name of function, class or object.
Furthermore, you get context help in Spyder after typing function name or namespace of interest. In practice, the fastest and easiest way to get help in Python is often to simply Google your problem. For instance: `"How to add legends to a plot in Python"` or the content of an error message. In the later case, it is often helpful to find the *simplest* script or input to script which will raise the error.

Discussion forum: You can get help on our online discussion forum:
Piazza forum: https://www.piazza.com/dtu.dk/spring2022/02450

1.1 How to do the exercises

During exercises in the course, you will go through an exercise document like this one. The exercises are centred around running and understanding a series of scripts provided in the 02450 Toolbox. The exercise descriptions guide you through the scripts and note that you won't have time to code everything from scratch. The scripts are also the basis for your work in the reports, where you will be able to re-use large parts of the code. However, for the reports, you will tailor the scripts to your dataset and problem.

The exercises are structured as smaller numbered sections. When a certain section concerns a particular script, it will be stated and their number will match. For instance, the first script you will run (in a little while) is called `ex1_5_1.py` and corresponds to the section 1.5.1 in this document.

1.2 Getting started with Python

We assume that you have a working Python IDE set up. If that is not the case, complete the pre-exercise (Exercise 0) before proceeding. If you have already done the optional Exercise 0, you can skip the next section ("Installing the 02450 Toolbox").

We will assume you have the most recent **Anaconda** Python distribution (Python 3.6). Additionally, in the following, it will be assumed Spyder is used to run python commands and edit Python files.

1.3 Installing the 02450 Toolbox

The course will make use of several specialized scripts and toolboxes not included with Python. These are distributed as a toolbox which need to be installed.

1.3.1 Download and unzip the 02450 Toolbox for Python, `02450Toolbox_Python.zip` . It will be assumed the toolbox is unpacked to create the directories:

```
<base-dir>/02450Toolbox_Python/Tools/     # Misc. tools and packages
<base-dir>/02450Toolbox_Python/Data/      # Datasets directory
<base-dir>/02450Toolbox_Python/Scripts/   # Scripts for exercises
```

For the exercises, you should work on the example scripts in `<base-dir>/02450Toolbox_Python/Scripts/` (notice the scripts are labelled according to exercise number) and not try to write the scripts from the bottom up.

1.3.2 To finalize the installation you need to update your path. In Spyder, go to `Tools -> PYTHONPATH Manager` and press `Add path`. Navigate to `<base-dir>/02450Toolbox_Python/Tools/` and press `Select folder`. Restart Spyder for changes to take effect. Test your path by typing:

```
import toolbox_02450
print(toolbox_02450.__version__)
```

If a revision with a date is printed the path to the toolbox is correctly set up.

1.4 Representation of data in Python

We will use a standard data representation throughout the course. Using this representation makes it easy to apply the various tools in the 02450 Toolbox on a new dataset. Once you have a given dataset in the standard format, the scripts will all be set up to work with it correctly.

An overview of the format is presented for Python in this table:

	Python var.	Type	Size	Description
	`X`	numpy.array	$N \times M$	Data matrix: The rows correspond to N data objects, each of which contains M attributes.
	`attributeNames`	list	$M \times 1$	Attribute names: Name (string) for each of the M attributes.
	`N`	integer	Scalar	Number of data objects.
	`M`	integer	Scalar	Number of attributes.
Classification	`y`	numpy.array	$N \times 1$	Class index: For each data object, `y` contains a class index, $\mathtt{y}_n \in \{0, 1, \dots, C-1\}$, where C is the total number of classes.
Classification	`classNames`	list	$C \times 1$	Class names: Name (string) for each of the C classes.
Classification	`C`	integer	Scalar	Number of classes.

1.5 Loading data

Before we can begin to do machine learning, we need to load the data. Datasets are distributed as various types of files, and a few common ones will be shown here for future reference to be used once you have to load your own dataset.

Once we have loaded a dataset, we often need to process the data before the format fits our needs. For this course, in particular, this mostly means putting the dataset in the $\mathbf{X}$, $\mathbf{y}$-format shown above. The machine learning algorithms you will use needs the data to be in a numerical format, so we will also go through how to convert data which is in a text format into a numerical format.

Lastly, we will also go through a few tasks that often need to be handled before we can load some dataset.

For today's exercises you need to add a package to your Python environment. The additional package is for importing data from excel spreadsheets. Please make sure that you have installed the following packages (you can follow the guidelines at the corresponding websites):

- Excel file data extraction (xlrd package):
  `https://xlrd.readthedocs.io`

The websites provide documentation of the packages. Note if you use the Anaconda Python distribution these packages may already be added, use `conda list` in the terminal for a list of installed packages.

For illustrating loading data, we will consider the Iris flower dataset, which we will also return to later on. The Iris flower dataset or Fisher's Iris dataset is a multivariate dataset introduced by Sir Ronald Aylmer Fisher (1936) for the problem of classifying Iris flower types. It is sometimes called Anderson's Iris dataset because Edgar Anderson collected the data to quantify the geographic variation of Iris flowers in the Gaspé Peninsula. The dataset consists of 50 samples from each of three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor). Four variables were measured from each sample, they are the length and the width of sepal and petal, in centimetres. Based on the combination of the four variables, Fisher developed a linear discriminant model to distinguish the species from each other. It is used as a typical test for many other classification techniques (see also `http://en.wikipedia.org/wiki/Iris_flower_data_set`). The data has been downloaded from `http://archive.ics.uci.edu/ml/datasets/Iris`.

The perhaps most common and simple format of storing data is the comma-separated values-file format (or CSV). In such files, the data is stored such that a sample or an observation is a line in a text document, and the document then has as many lines (or rows) as there are samples. The attribute values for an observation is written within one line, separated by (usually) a comma or a tab-character in a consistent order. This order is usually defined in a header (the first line of the file), which has a designation of the variable name in some format.

1.5.1 Inspect the file

`<base-dir>/02450Toolbox_Python/Data/iris.csv` using a simple text editor (e.g. for Windows "Notepad" or for MacOS "TextEdit"). Afterwards, inspect the script `ex1_5_1.py` to see how to load the Iris data from a CSV-file and put it into the standard format. Since the class label (the flower species) are stored as text (or strings), we convert them into a numerical value.

1.5.2 Sometimes datasets are distributed as Excel-files (`.xls(x)`). Inspect the script `ex1_5_2.py` to see how to load the same Iris data, when it has been stored as an Excel-file (open

`<base-dir>/02450Toolbox_Python/Data/iris.xls`

to have a look at the file).

1.5.3 Other times, and especially in this course, data is stored as MATLAB files (`.mat`). Inspect `ex1_5_3.py` to load the Iris data from

`<base-dir>/02450Toolbox_Python/Data/iris.mat` .

1.5.4 In the examples up until now, we have handled the data in the Iris dataset as if to solve a classification problem. We could say that the *primary* machine learning modelling aim is to classify the species of Iris flower based on the petal and sepal dimensions. However, we could also use the dataset to illustrate how to do regression without needing to use a whole different dataset. We would achieve this by e.g. trying to predict either of the petal (or sepal) dimensions based on the remaining dimensions, for instance. This changes how we define our $\mathbf{X}$,$\mathbf{y}$-format. Inspect `ex1_5_4.py` to see how to cast the Iris dataset into a regression problem. In the script, we will set up the $\mathbf{X}$,$\mathbf{y}$-format such that we are predicting the petal lengths form the other available information. Notice that we change how we use the information of the class label from before (the species information). Instead of storing it as a single variable, we have now used a "one-out-of-K encoding", since it is a categorical variable—we will return to various types of variables when we go through chapter 2 in the book (where one-out-of-K-encodings are described in section 2.4.1).

1.5.5 While the Iris dataset is a real dataset, it is a very clean and easy to work with dataset. Usually, data is a bit messier, and we will consider a toy dataset that has some common issues. Often, the description of "real-world" data is stored along with the data in some form of a text file. Have a look at the folder

`<base-dir>/02450Toolbox_Python/Data/messy_data/`,

and read more about the toy dataset—notice that there is a `README.txt` file.

Inspect the data in `messy_data.data` and try to identify some issues (use e.g. simple text editor as before). Afterwards, inspect `ex1_5_5.py` to see how the dataset can be stored in the desired representation.

1.6 Select a dataset for the reports

The course will include two written group reports. The two reports will cover the first two sections of the course:

- Data: Feature extraction, and visualization
- Supervised learning: Classification and regression

Each group will find one dataset they will use for the reports. This can either be your own dataset or a dataset you find yourself. Additional details about where you can find a dataset and formal requirements can be found in the **Finding a dataset for the reports** `.pdf` file on DTU Learn.

As a guideline, your dataset should have at least 60 observations and 5 attributes with at least two of the attributes being interval or ratio. Once you have found a dataset you need to have the dataset approved by one of the teaching assistants of the course.

As part of project 1, you will have to think about how various machine-learning tasks can be carried out for the dataset chosen and this exercise will briefly touch upon this. Discuss the following questions:

1.6.1 What is the problem of interest? (describe what your data is about in general terms, i.e. "to someone who knows nothing of machine learning")

1.6.2 Who made the data and why? Did they, or somebody else, work with the data and report results? If so, what were their results?

1.6.3 What is the *primary* machine learning modelling aim? (Is it *primarily* a classification, a regression, a clustering, an association mining, or an anomaly detection problem?)

1.6.4 Which attributes are relevant when carrying out a classification, a regression, a clustering, an association mining, and an anomaly detection? Specifically: Which attribute do you wish to explain in the regression based on which other attributes? Which class label will you predict based on which other attributes in the classification task?

1.6.5 Are there any data issues? Either directly reported in the accompanying dataset description or apparent by inspection of the data? (such as missing values or incorrect/corrupted values)

1.7 Tasks for the report

You are now able to address the following tasks for the report

1. **A description of your data set.**
   - Explain what your data is about. I.e. what is the overall problem of interest?
   - Provide a reference to where you obtained the data.
   - Summarize previous analysis of the data. (i.e. go through one or two of the original source papers and read what they did to the data and summarize their results).
   - You will be asked to apply (1) classification and (2) regression on your data in the next report. For now, we want you to consider how this should be done. Therefore:
     Explain, in the context of your problem of interest, what you hope to accomplish/learn from the data using these techniques?.
     Explain which attribute you wish to predict in the regression based on which other attributes? Which class label will you predict based on which other attributes in the classification task?
     If you need to transform the data in order to carry out these tasks, explain roughly how you plan to do this.

One of these tasks (1)–(5) is likely more relevant than the rest and will be denoted the **main machine learning aim** in the following. The purpose of the following questions, which asks you to describe/visualize the data, is to allow you to reflect on the feasibility of this task.

References