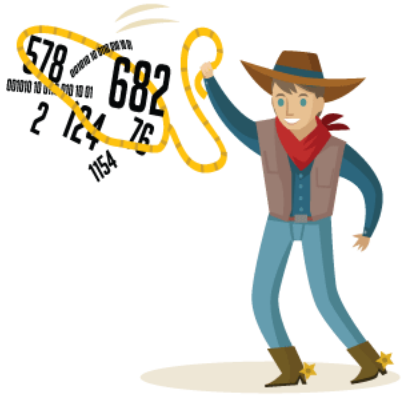
Data Processing: Readin', Ritin', and Rithmetic
ML_2022: Machine Learning

https://people.sc.fsu.edu/~jburkardt/classes/ml_2022/data_lab/data_lab.pdf



You've got to rope in your data before you can process it.

Processing Data!

- *We will download several datafiles from the class website;*
- *These datafiles will be of* **txt** *and* **csv** *formats;*
- *The* **txt** *files are purely numeric, but* **csv** *files may contain other kinds of data;*
- *To extract data, we will use* `loadtxt()` *and the* `pandas()` *function* `read_csv()`*;*
- *We will compute data statistics by calling functions like* `np.min()`*;*
- *We will practice normalizing or standardizing the data;*
- *We will plot a component (column) of a dataset;*
- *We will deal with missing values in a dataset;*

1 Getting data

Each of the exercises will be carried out on a particular datafile. These datafiles are available on the *datasets* page at the class website:
https://people.sc.fsu.edu/~jburkardt/classes/ml_2022/datasets/datasets.html

You might go ahead now and download them all:

- *homes_data.txt*
- *wine.csv*
- *turtles.csv*
- *diabetes.csv*

2 Exercise 1:

Write a program `exercise1.py` and:

- use `np.loadtxt()` to read `data` from *homes_data.txt*;
- use `np.shape()` to get and print the number of rows and columns;
- print the first five rows of `data`;
- compute and print the minimum, maximum, range, mean, variance of `data`;
- create `data2`, a normalized copy of `data`;
- compute and print the minimum, maximum, range, mean, variance of `data2`;

There is no `numpy()` function to compute the range of an array; simply compute the difference of the maximum and minimum values.

3 Prepare for Exercise 2:

Now we will be reading `csv` files, so we will need the statement

```
import pandas as pd
```

and we will have to learn a little about *dataframes*. A dataframe is similar to a `numpy()` array. It is essentially a table of data, and we read it from a file using a command like:

```
df = pd.read_csv ( filename )
```

As with arrays, an individual entry is `df[i,j]`, a row is `df[i,:`, and a column is `df[:,j]`. As it happens, this datafile does not include an initial header line,] and all the data is numeric. Therefore, now that we've extracted the information into `df`, we can simply request that it be converted into the kind of `numpy()` array with which we are familiar:

```
data = df.to_numpy ( )
```

and after this is done, there are no more surprises!

After printing the usual statistics for `data`, you will now be asked to apply standardization, using the mean and variance of the data.

4 Exercise 2:

Write a program `exercise2.py` and:

- use `pd.read_csv()` to read `df` from *wines.csv*;
- use `df.to_numpy()` to create the `numpy()` array `data`;
- use `np.shape()` to get and print the number of rows and columns;
- print the first five rows of `data`;
- compute and print the minimum, maximum, range, mean, variance of `data`;
- create `data2`, a standardized copy of `data`;
- compute and print the minimum, maximum, range, mean, variance of `data2`;

5 Prepare for Exercise 3

The `csv` format is used because it allows us to include an optional header line describing each column of data, and it allows non-numeric data items such as strings. In our next example, we will try to deal with such data. We will read the file in the usual way, print out the first five rows, and realize we are dealing with a mixture of numeric and string data.

We will then copy the numeric data into a `numpy()` array, by listing the headers of the columns we are interested in, and proceed from there.

6 Exercise 3:

Write a program `exercise3.py` and:

- use `pd.read_csv()` to read `df` from *turtles.csv*;
- print the first five rows of `df` using this "crazy" command:

```
for i in range ( 0, 5 ):
  print ( df.loc[i] )
```

- use a version of the command `df.to_numpy()` to create the `numpy()` array `data` from the numeric columns;

```
data = df[['Length','Width','Height']].to_numpy ( )
```

- use `np.shape()` to get and print the number of rows and columns in `data`;
- print the first five rows of `data` (the usual way!);
- use `plt.hist ( data[:,0], bins = 20 )` to make a histogram of turtle lengths;
- use `plt.savefig('exercise3.jpg')` to save your plot;
- compute and print the minimum, maximum, range, mean, variance of `data`;

7 Prepare for Exercise 4

The `csv` datafile *diabetes.csv* contains an initial header line, but all the data is numeric. However, in many cases, a given data value was not available, and so instead a zero value was entered. This is a case of "missing data". In some cases, a record containing missing data is still usable, but we will assume that any such record is useless, and we will want to eliminate such records before proceeding.

pandas() has two useful procedures that we will call:

- `df.replace(old,new)` replaces values of `old` by the value `new`;
- `df.dropna()` drops from `df` all records with any `NaN` values;

We only want to do the zero replacement in certain columns, so to do so, instead of specifying the simple `df.replace()` command, we will use `df[[header0,header1,...,headerk]].replace()` where `header1` is the header of the first column we want to modify, and so on.

The headers for *diabetes.csv* are

0. *"Pregnancies"*
1. *"Glucose"*
2. *"Diastolic"*
3. *"Triceps"*
4. *"Insulin"*
5. *"BMI"*
6. *"Pedigree"*
7. *"Age"*
8. *"Class"*

and we will want to operate on columns *"Glucose"*, *"Diastolic"*, *"Triceps"*, *"Insulin"*, and *"BMI"*.

Once we have cleaned up the data, we will want to make a histogram of column *"Triceps"*, and then compute and print the data statistics.

8 Exercise 4:

Write a program `exercise4.py` and:

- use `pd.read_csv()` to read `df` from *diabetes.csv*;
- print a bit of the file:

```
print ( df.describe() )
```

- The headers are stored as `df.columns` and we can print all of them:

```
for col in df.columns:
  print ( col )
```

- Count the number of 0 values in columns 1, 2, 3, 4, and 5:

```
missing = (df[["Glucose","Diastolic","Triceps","Insulin","BMI"]]==0).sum()
print ( missing )
```

- In columns 1, 2, 3, 4, 5, replace any 0 value by NaN:

```
df[["Glucose","Diastolic","Triceps","Insulin","BMI"]] = df[["Glucose","Diastolic","
    Triceps","Insulin","BMI"]].replace(0, np.nan)
```

- Now drop every record containing at least one NaN value:

```
df.dropna ( inplace = True )
```

- Use `df.to_numpy()` to create the `numpy()` array `data`:

```
  data = df.to_numpy ( )
```

- use `np.shape()` to get and print the number of rows and columns in `data`;
- use `plt.hist ( data[:,3], bins = 20 )` to make a histogram of the *"Triceps"* data;
- use `plt.savefig('exercise4.jpg')` to save your plot;
- compute and print the minimum, maximum, range, mean, variance of `data`;

.