**# Introduction**

This Documentation file covers what is Fisher's Iris Dataset, why it is used in data analysis, methods for analysing data in Python and then examples of how this can be done with the Fisher's Iris Dataset.

**# The Fisher's Iris Dataset**

**## What is Fisher's Iris Dataset?**

The Fisher's Iris dataset is a simple dataset of 150 records: 50 records of 3 different types of irises. There are 5 variables in the dataset:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:

* Iris Setosa
* Iris Versicolour
* Iris Virginica

The data was collected by Edgar Anderson who was trying to work out how one species of Iris evolved from another. He chose to study Iris Versicolour, but subsequently discovered this was actually two species: Iris Versicolour and Iris Virginica. He added to his investigation Iris Setosa. From his investigation he discovered that the Iris Versicolour in North East America as a breeding of Iris Viginica and Irish Setosa.

The dataset was published by Ronald Fisher in his 1936 paper The use of multiple measurements in taxonomic problems [1]. From analysing the data Fisher showed that Iris Setosa could be separated from Iris Versicolour and Iris Virginica. Through this project we are going to use Python to create some visual representations of the data that confirms this differentiation.

**## Where can it be located?**

The data set was donated in 1988 by Michael Marshall and is published to the public at UCI Machine Learning Repository [2].

**## What are its uses?**

**# Analysing data with Python**

**## Working with numerical data: numpy**

Numpy is a library designed for working with numerical data. You first need to ensure it is installed on your computer and then import it in your code.

**## Working with datasets: PANDAS**

The most commonly used tool in Python for working with data sets is Pandas [3]. It is a Python library with functions for analysing, cleaning, exploring and manipulating data. The codebase for Pandas is available at: <https://github.com/pandas-dev/pandas>.

Before using Pandas it will need to be installed. If you already have Spyder or Anaconda installed this already contains Pandas and you don't need to install it again.

Once it is installed it is added to my applications by importing it, this is normally done under the alias pd:

​

An easy each way to check if it is installed is to request the version number to be printed to the terminal:

1.4.4

Pandas allows tabulated data such as csv to be imported as a data table, in Pandas this is called DataFrame. Pandas allows the importing of data from a variety of file formats or data sources using the format read\_ source type, eg read\_csv will read in data from a csv file. By default the first line of the file is assumed to be the header file. If the file does not have headers these can added by using names = " ", " "].

A quick view of the data can be seen by printing head() or info(). Here I have used info() to confirm the column names, the number of entries in each column and the datatype.

By default the output will show upto 60 lines, if these is more than this it will only show the first and last five. The max rows for your setup can be checked with pd.options.display.max\_rows. The maximum number of rows can be increased by using the same statement and suffixing = and then what ever number you want. This is essential if you would like to print the entire dataframe to the terminal to review the data, although for large datasets a text editor is normally better suited to this task.

The above information can be used to identify columns with null values and the user can make decisions how to handle these, for example, the whole row could be removed or the missing value replaced with the mean for the column. Fisher's Iris data set is not missing any values. Another useful test is to check for duplicated values using the duplicated().

Within Pandas data cleaning can take place. It has methods to clean empty cells, change cell format, clean incorrect data and remove duplicates.

When the data is imported you can filter on a condition, or select by row and/or column.The loc attribute returns a specific row or rows.

Within Pandas basic statistics about a column (series ) of data can be calculated using the format

**## Plotting data**

Data visualisation helps makes complex data sources more easy to understand via graphics. Matplotlib and Seaborn are the most commonly used data visualisation tools in Python.

**### Matplotlib**

Matplotlib is a library for the plotting of data. It works alongside Pandas and Numpy. The codebase for Matplotlib is available at <https://github.com/matplotlib/matplotlib>. The figures can be interactive, the plots can be formatted and exported into a variety of formats. Matplotlib is normally imported under the alias 'plt'. Most of the Matplotlib utilities are in the pyplot sublibrary and therefore, it is often imported at this level.

**### Seaborn**

Seaborn is also a Python library and it can make more complicated plots. It is basically a more complicated overlay on matplotlib. There is extensive information available at: <https://seaborn.pydata.org/index.html>

**### Matplotlib or Seaborn: Which is better?**

As is often the case with this question, the answer is, it depends on what you are trying to do. Matplotlib is good at making basic graphs including barcharts, piecharts and scatterplots. Seaborn is an extended version of matplotlib and uses matplotlib, pandas and numpy to make visualisations. The following table is based on the one at: <https://www.geeksforgeeks.org/difference-between-matplotlib-vs-seaborn/>.

| **Area** | **Matplotlib** | **Seaborn** |
| --- | --- | --- |
| Function | Basic charts including bar charts, pie charts and scatter plots | More complicated plots. It can provide a plot of all data one plot. |
| Syntax | Is quite lengthy eg matplotlib.pyplot.hist(x\_axis, y\_axis) | Is a little simpler and easier to learn eg seaborn.histplot(x\_axis, y\_axis) |
| Multiple plots | Can open and use multiple plots at a time. Need to be closed. | Time set for the creation of each plot, this can lead to out of memory errors |
| Visualisation | Provides similar output and syntax as MATLAB, so great for those already familiar with MATLAB | More comfortable at handling Pandas dataframes. Great variety of nice looking visualisations can be created |
| Data frames and arrays | Works efficiently with data frames and arrays. Plot() can be called without parameters. | Whole data set is treated as a simple unit and parameters are needed when calling plot(). |

**# How to run the code**

**## Explaining the repository**

The repository on github (<https://github.com/kknb1982/pands-project>) includes both the code and the output file. The code and how the file contents are created are described in later sections of this notebook, but here is a brief explanation of each file:

* Analysis.py: This is the file to run to create the dataframe and plots
* Petal Length.png: This is a histogram of the Petal Length data split by Species of Iris
* Petal Width.png: This is a histogram of the Petal Width data split by Species of Iris
* Sepal Length.png: This is a histogram of the Sepal Length data split by Species of Iris
* Sepal Width.png: This is a histogram of the Sepal Width data split by Species of Iris
* combinedhist.png: This fiel contains histograms of the variable data it is not segregated by species
* pairplot.png: This file uses the Seaborn parplot function to output multiple scatterplots of two variables. The plots are coloured by species.
* summary.txt: Gives a summary of each variable
* variablemodule.py: Contains all the code for the functions and importing the dataset

**## Importing the dataset**

The Fisher's Iris data set is published at <https://archive.ics.uci.edu/ml/datasets/iris>. The CSV file for the data is at <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data> and the details for the headers is in <https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.names>. To import the dataset I am using Pandas as first I need to import the Pandas library. I have used the alias pd.

The source dataset does not include the headers, so we need to create these.

datafields **=** sepallen, sepalwid, petallen, petalwid, species

When importing the dataset we add the argument names= to add the correct header data. Our dataset is in a

**# Analysing Fisher's Iris Dataset**

**## Summary text file**

A summary of each variable to a single text file

**## Histogram of each variable**

There is a greate resource at : <https://realpython.com/python-histograms/> all about the different methods of creating histograms in Python.

**### Create simple histogram**

First I created a simple histogram using the Pandas hist() method. Thsis automatically creates a histogram of each variable and outputs at one time. From these plots you can see that Sepal Width appears to have normal type distribution a

Chart, histogram

Description automatically generated

In order to save the combined plot to the file, we first need to install matplotlib.pyplot. I imported this under the alias plt as is common practice. The command to save the figure rather than output to the terminal is plt.savefig().

**### Histogram showing species**

Next I used the Seaborn library to create a more complicated histogram showing the species of the Iris. To do this import seaborn

**import** seaborn **as** sns

Then there is a for loop to work through each variable in turn. There is a nested if loop to ignore the species column as this is not numerical.

The format for creating a histplot in seaborn is sns.histplot(). Parameters can be added to the call for a plot, there is great information about this at: <https://seaborn.pydata.org/generated/seaborn.histplot.html?highlight=hist#seaborn.histplot>

**for** name **in** datafields:

**if** name **!=** species:

sns.histplot(data**=**dataf, x**=**name, hue**=**species, binwidth**=**0.5)

plt.savefig(name**+** '.png')

Type *Markdown* and LaTeX: 𝛼2�2

Historgrams to be saved to the repository <https://www.geeksforgeeks.org/how-to-create-a-histogram-from-pandas-dataframe/>

**## Scatterplots of pairs of variables**

To be saved in the repository

**## examples of interesting analyses that others have pursued based on**

the data set will be discussed

**# References**

[1] Fisher paper

<https://github.com/pandas-dev/pandas>