**# Table of contents**

**# 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. There is a separate Readme file that describes the final version of my code.

**# 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. The data was collected by Edgar Anderson who was trying to work out how one species of Iris evolved from another(1,2). He chose to study Iris Versicolour and through his word he discovered that this was actually two species: Iris Versicolour and Iris Virginica. He also studied Iris Setosa and from his investigation he discovered that the Iris Versicolour in North East America is a breeding of Iris Virginica and Irish Setosa. To complete this research Anderson collected 5 variables about example of the different irises:

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 simplicity of the dataset and its relatively small size makes it ideal for statistical analysis, computer science and machine learning. The dataset was famously used by Ronald Fisher in his 1936 paper The use of multiple measurements in taxonomic problems (3), hence it being commonly known as Fisher’s Iris Dataset. 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 (4).

**## What are its uses?**

The Fisher’s Iris Dataset is small, with limited variables and categories (classes of iris), this makes it a great place to test statistical, computer science and machine learning techniques. Fisher’s 1936 paper (5) publishing the dataset has been referenced in 8,702 academic papers and sub-set of these use the dataset (rather than the statistical techniques he discovered in the paper) for everything from designing artificial intelligence classifiers (6) to testing techniques for handling datasets with missing values (7).

**# Analysing data with Python**

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

Numpy is a library designed for working with numerical data (8). 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 (9). 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 (10). Pandas allows the importing of data from a variety of file formats or data sources using the format read\_ source type, eg read\_csv (10)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 (11).

**### Matplotlib**

Matplotlib is a library for the plotting of data(12). 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 (13) and it can make more complicated plots(14). 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/> (15).

| **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:

* Readme: Describes the final version of the code in brief
* 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 file 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> (4). 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(9). 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

(16)

**# 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

(17–20)

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().

(21–25)

(26)

(27–29)

(30)

(31)**### 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

(32,33)

**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().

(34)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 o**

(35,36)**f variables**

To be saved in the repository

**## ex**

the data set will be discussed

**# References**

(37)

[1] Fisher paper

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

(4,5,9–12,38–40)

1. Anderson E. The Species Problem in Iris. Annals of the Missouri Botanical Garden. 1936 Sep;23(3):457.

2. E. A. The IRISes of the Gaspe peninsula. Bulletin American IRIS Society [Internet]. 1935 [cited 2023 May 4];39:2–15. Available from: https://cir.nii.ac.jp/crid/1574231874415574912

3. FISHER RA. THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS. Ann Eugen [Internet]. 1936 Sep 1 [cited 2023 May 4];7(2):179–88. Available from: https://onlinelibrary.wiley.com/doi/full/10.1111/j.1469-1809.1936.tb02137.x

4. UCI Machine Learning Repository: Iris Data Set [Internet]. [cited 2023 Apr 18]. Available from: https://archive.ics.uci.edu/ml/datasets/iris

5. FISHER RA. THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS. Ann Eugen [Internet]. 1936 Sep [cited 2023 Apr 18];7(2):179–88. Available from: https://onlinelibrary.wiley.com/doi/10.1111/j.1469-1809.1936.tb02137.x

6. AETiC >> Archive [Internet]. [cited 2023 May 8]. Available from: http://aetic.theiaer.org/archive/v7/v7n1/p5.html

7. Poudevigne-Durance T, Jones OD, Qin Y. MaWGAN: A Generative Adversarial Network to Create Synthetic Data from Datasets with Missing Data. Electronics 2022, Vol 11, Page 837 [Internet]. 2022 Mar 8 [cited 2023 May 8];11(6):837. Available from: https://www.mdpi.com/2079-9292/11/6/837/htm

8. NumPy [Internet]. [cited 2023 May 4]. Available from: https://numpy.org/

9. Categorical data — pandas 2.0.1 documentation [Internet]. [cited 2023 Apr 24]. Available from: https://pandas.pydata.org/pandas-docs/stable/user\_guide/categorical.html

10. IO tools (text, CSV, HDF5, …) — pandas 2.0.0 documentation [Internet]. [cited 2023 Apr 18]. Available from: https://pandas.pydata.org/pandas-docs/stable/user\_guide/io.html#csv-text-files

11. Python Histogram Plotting: NumPy, Matplotlib, pandas & Seaborn – Real Python [Internet]. [cited 2023 Apr 22]. Available from: https://realpython.com/python-histograms/

12. Matplotlib — Visualization with Python [Internet]. [cited 2023 Apr 18]. Available from: https://matplotlib.org/

13. Waskom M. seaborn: statistical data visualization. J Open Source Softw. 2021 Apr 6;6(60):3021.

14. Waskom M. seaborn: statistical data visualization. J Open Source Softw. 2021 Apr 6;6(60):3021.

15. Difference Between Matplotlib VS Seaborn - GeeksforGeeks [Internet]. [cited 2023 May 8]. Available from: https://www.geeksforgeeks.org/difference-between-matplotlib-vs-seaborn/

16. Get unique values from a column in Pandas DataFrame - GeeksforGeeks [Internet]. [cited 2023 May 8]. Available from: https://www.geeksforgeeks.org/get-unique-values-from-a-column-in-pandas-dataframe/

17. Writing to file in Python - GeeksforGeeks [Internet]. [cited 2023 May 8]. Available from: https://www.geeksforgeeks.org/writing-to-file-in-python/

18. Python For Loops [Internet]. [cited 2023 May 8]. Available from: https://www.w3schools.com/python/python\_for\_loops.asp

19. Python Conditions [Internet]. [cited 2023 May 8]. Available from: https://www.w3schools.com/python/python\_conditions.asp

20. Python Functions [Internet]. [cited 2023 May 8]. Available from: https://www.w3schools.com/python/python\_functions.asp

21. pandas.DataFrame.min — pandas 2.0.1 documentation [Internet]. [cited 2023 May 8]. Available from: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.min.html

22. pandas.DataFrame.mode — pandas 2.0.1 documentation [Internet]. [cited 2023 May 8]. Available from: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.mode.html

23. pandas.DataFrame.median — pandas 2.0.1 documentation [Internet]. [cited 2023 May 8]. Available from: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.median.html

24. pandas.DataFrame.mean — pandas 2.0.1 documentation [Internet]. [cited 2023 May 8]. Available from: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.mean.html

25. pandas.DataFrame.max — pandas 2.0.1 documentation [Internet]. [cited 2023 May 8]. Available from: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.max.html

26. Python Escape Characters [Internet]. [cited 2023 May 8]. Available from: https://www.w3schools.com/python/gloss\_python\_escape\_characters.asp

27. How to Write to Text File in Python [Internet]. [cited 2023 May 8]. Available from: https://www.pythontutorial.net/python-basics/python-write-text-file/

28. Python 3’s f-Strings: An Improved String Formatting Syntax (Guide) – Real Python [Internet]. [cited 2023 May 8]. Available from: https://realpython.com/python-f-strings/

29. pandas.Series.describe — pandas 2.1.0.dev0+735.gf1126610ab documentation [Internet]. [cited 2023 May 8]. Available from: https://pandas.pydata.org/pandas-docs/dev/reference/api/pandas.Series.describe.html#pandas.Series.describe

30. pandas.DataFrame.to\_string — pandas 2.0.1 documentation [Internet]. [cited 2023 May 8]. Available from: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.to\_string.html

31. Strings and Character Data in Python – Real Python [Internet]. [cited 2023 May 8]. Available from: https://realpython.com/python-strings/

32. Histograms — Matplotlib 3.7.1 documentation [Internet]. [cited 2023 May 8]. Available from: https://matplotlib.org/stable/gallery/statistics/hist.html

33. matplotlib.pyplot.savefig — Matplotlib 3.7.1 documentation [Internet]. [cited 2023 May 8]. Available from: https://matplotlib.org/stable/api/\_as\_gen/matplotlib.pyplot.savefig.html#matplotlib.pyplot.savefig

34. seaborn.histplot — seaborn 0.12.2 documentation [Internet]. [cited 2023 May 8]. Available from: https://seaborn.pydata.org/generated/seaborn.histplot.html?highlight=histplot

35. matplotlib.pyplot.xlabel — Matplotlib 3.7.1 documentation [Internet]. [cited 2023 May 8]. Available from: https://matplotlib.org/stable/api/\_as\_gen/matplotlib.pyplot.xlabel.html

36. matplotlib.pyplot.title — Matplotlib 3.7.1 documentation [Internet]. [cited 2023 May 8]. Available from: https://matplotlib.org/stable/api/\_as\_gen/matplotlib.pyplot.title.html

37. Ursani Z, Ursani AA. The Theory of Probabilistic Hierarchical Learning for Classification. Annals of Emerging Technologies in Computing [Internet]. 2023 Jan 1;7(1):61–74. Available from: http://aetic.theiaer.org/archive/v7/v7n1/p5.html

38. seaborn.pairplot — seaborn 0.12.2 documentation [Internet]. [cited 2023 Apr 22]. Available from: https://seaborn.pydata.org/generated/seaborn.pairplot.html

39. python - Save plot to image file instead of displaying it - Stack Overflow [Internet]. [cited 2023 Apr 22]. Available from: https://stackoverflow.com/questions/9622163/save-plot-to-image-file-instead-of-displaying-it

40. How do I select a subset of a DataFrame? — pandas 2.0.0 documentation [Internet]. [cited 2023 Apr 22]. Available from: https://pandas.pydata.org/pandas-docs/stable/getting\_started/intro\_tutorials/03\_subset\_data.html

The next module creates scatter plot for the variable pairs and histograms for the univariate plots.

createpairplot()

The next module creates box plots to graphically show most of the statistical data from the text file.

getboxplots()

The final module creates violinplots which shows the distribution of the data across the range.

getviolinplots()

### 3.2.9 Create boxplots

Boxplots are very useful for giving a visual representation of the statistical information about some data. I used the Pandas Boxplot method to create the plots [[30]](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.boxplot.html). The default figure sizing meant that the species labels overlapped making them unreadable, so I increased it to 11 by 11. Again, I used savefig [[25]](https://matplotlib.org/stable/api/\_as\_gen/matplotlib.pyplot.savefig.html#matplotlib.pyplot.savefig) to save a copy of this figure to the repository (https://github.com/kknb1982/pands-project/blob/main/boxplot.png).

![image](https://github.com/kknb1982/pands-project/assets/123597290/a1f73895-9758-41d9-bf83-7429032fde22)

Copied from https://aiaspirant.com/box-plot/

def getboxplots():

dataf.boxplot(by=species, figsize=(11,11))

plt.savefig('boxplot.png')

plt.close()

### 3.2.10 Create violinplots

A violin plot shows the distribution of data in a unique way [[31]](https://chartio.com/learn/charts/violin-plot-complete-guide/). I used a `for` loop to create each violinplot in turn [[10]](https://www.w3schools.com/python/python\_for\_loops.asp) and an `if` statement to ensure graphs for all variables bar the species were created [[11]](https://www.w3schools.com/python/python\_conditions.asp). To create the violinplot I used Seaborn [[32]](https://seaborn.pydata.org/generated/seaborn.violinplot.html). Defining the `x` parameter as species, splits the data by species in each of the plots.

def getviolinplots():

for name in datafields:

if name != species:

sns.violinplot(data=dataf, x=species, y=name)

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

plt.close()