**PART 1**

**PYTHON APPLICATION SUMMARY**

Over the next few pages I shall briefly summarize the twelve menu sections of a simple Python program I designed called Analysis, which can be used to interrogate the Iris dataset.

The Python application contained within this project isn’t so much a demonstration of my understanding of the Iris Data Set, but serves more of an introduction to the users of what can be done with some of the basic commands and modules that are available for use with Python that can assist further analysis.

|  |
| --- |
| For composing the main menu for this project, I found Dr Gerard Harrison’s file useful as a clean template to modify for my purposes.  <https://learnonline.gmit.ie/mod/resource/view.php?id=81195> |

There were a few sources which enabled me to complete the Python application. Prior to doing this assignment I had trained for an MTA in Python with ecollege, so already had an extensive amount of practice in some basic areas, hence my references to external sources may be light.

**MENU ITEMS OF ANALYSIS.PY**

**1 – Introduction To Iris Data Set**

* This section is simple print summary about the Iris Data Set
* It is followed by a very simple description about the Python application.

**2 - View Image Of Iris Varieties**

* This section is a simple demonstration that Python can load images
* It shows the user the code for how this can be done.

e.g. CODE USED TO DISPLAY AN IMAGE WITH PYTHON

*from PIL import Image*

*image = Image.open('test.png')*

*image.show()*

<https://stackoverflow.com/questions/35286540/display-an-image-with-python>

Python, just like Excel, has the full range of inbuilt aggregate commands. In this Python Application I provide a cursory demonstration of how these aggregate functions when combined with Pandas library are coded to get results straight from the CSV file. These are demonstrated in menu items 3,4,5.

**3 – View Average Sizes Iris**

* This section is a simple demonstration that the library Pandas is very effective in extracting the Average / Mean of the data.
* It shows the user the code for how this can be done
* e.g. CODE USED TO GET THE MINIMUM SIZE OF SETOSA

*# importing pandas as pd*

*import pandas as pd*

*# Creating the dataframe*

*iris = pd.read\_csv('IRIS.csv')*

*#Used for filtering type column*

*iris\_setosa = iris.loc[iris['type'] == 'Setosa']*

*#Used for calculating averages*

*minimum\_of\_Setosa = iris\_setosa.min(axis = 0, skipna = True)*

*print("Minimum of Setosa")*

*print(pd.DataFrame(minimum\_of\_Setosa))*

Pandas Dataframe

<https://cmdlinetips.com/2018/02/how-to-subset-pandas-dataframe-based-on-values-of-a-column/>

<https://cmdlinetips.com/2019/10/pandas-groupby-13-functions-to-aggregate/>

<https://www.geeksforgeeks.org/python-pandas-dataframe-mean/>

**4 – View Minimum Sizes Iris**

* This section is a simple demonstration that the library Pandas is very effective in extracting the Minimum of the data.
* It shows the user the code for how this can be done
* e.g. CODE USED TO GET THE MINIMUM SIZE OF SETOSA

*# importing pandas as pd*

*import pandas as pd*

*# Creating the dataframe*

*iris = pd.read\_csv('IRIS.csv')*

*#Used for filtering type column*

*iris\_setosa = iris.loc[iris['type'] == 'Setosa']*

*#Used for calculating averages*

*minimum\_of\_Setosa = iris\_setosa.min(axis = 0, skipna = True)*

*print("Minimum of Setosa")*

*print(pd.DataFrame(minimum\_of\_Setosa))*

Pandas Dataframe

<https://cmdlinetips.com/2018/02/how-to-subset-pandas-dataframe-based-on-values-of-a-column/>

<https://cmdlinetips.com/2019/10/pandas-groupby-13-functions-to-aggregate/>

<https://www.geeksforgeeks.org/python-pandas-dataframe-mean/>

**5 – View Maximum Sizes Iris**

* This section is a simple demonstration that the library Pandas is very effective in extracting the Maximum of the data.
* It shows the user the code for how this can be done
* e.g. CODE USED TO GET THE MAXIMUM SIZE OF ALL IRIS

*# importing pandas as pd*

*import pandas as pd*

*# Creating the dataframe*

*iris = pd.read\_csv('IRIS.csv')*

*# sum over the column axis.*

*maximum\_of\_data = iris.max(axis = 0, skipna = True)*

*print("Maximum Sizes of All Iris Data")*

*print(maximum\_of\_data)*

Pandas Dataframe

<https://cmdlinetips.com/2018/02/how-to-subset-pandas-dataframe-based-on-values-of-a-column/>

<https://cmdlinetips.com/2019/10/pandas-groupby-13-functions-to-aggregate/>

<https://www.geeksforgeeks.org/python-pandas-dataframe-mean/>

**-----------------------------------**

**6 – Save Summary Data To Text File**

* This saves the data from the Pandas dataframe summaries of Average, Minimum and Maximum sizes to a text file.

<https://stackoverflow.com/questions/5214578/print-string-to-text-file>

**-----------------------------------**

**7 – View Paired Graph Plots**

* This demonstrates to the user, the functionality from Seaborne / Matplotlib Paired Plots

**8 – View Scatter Plots**

* This demonstrates to the user, the functionality from Seaborne / Matplotlib Scatter Plots

**9 – View Histograms**

* This demonstrates to the user, the functionality from Seaborne / Matplotlib Histograms

Seaborn and Matplotlib Plots

* <https://web.microsoftstream.com/video/025ef713-d7c8-492f-97f4-5590015da029>
* <https://seaborn.pydata.org/generated/seaborn.scatterplot.html>
* <https://stackoverflow.com/questions/50091591/plotting-seaborn-heatmap-on-top-of-a-background-picture>
* <https://medium.com/@avulurivenkatasaireddy/exploratory-data-analysis-of-iris-data-set-using-python-823e54110d2d>
* <https://stackoverflow.com/questions/51400076/change-seaborn-pair-plot-figure-size>

**-----------------------------------**

**10 – Save Paired Graph Plots**

* This saves a Paired Plot graph for the user
* It also uses the os module to show the user the current working directory of where the file is saved to

<https://web.microsoftstream.com/video/025ef713-d7c8-492f-97f4-5590015da029>

<https://seaborn.pydata.org/generated/seaborn.pairplot.html>

**11 – Save Scatter Plots**

* This saves all variations of scatter plots for the user
* It also uses the os module to show the user the current working directory of where the files are saved to

**12 – Save Histograms**

* This saves all variations of histograms for the user
* It also uses the os module to show the user the current working directory of where the files are saved to

Saving graph plots

* <https://web.microsoftstream.com/video/737a2b0f-5e2e-4831-a020-5e5d3f8a05e9>

**-----------------------------------**

**x – Exit application**

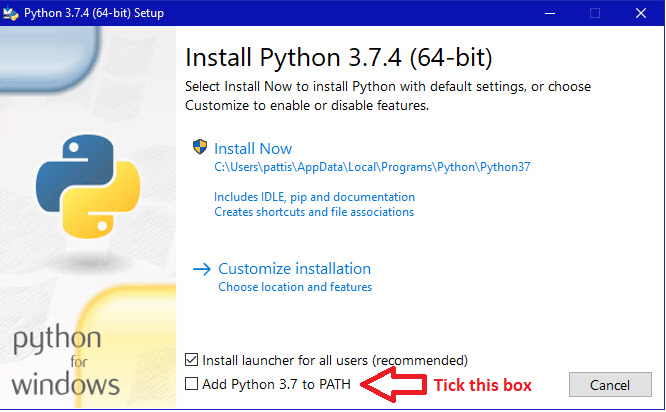
**SETUP**

To view screens in action please watch video demo I have provided at <https://youtu.be/tP9lVF4gXQs> which demonstrates downloading the folder from my github <https://github.com/g00387822/pands-project>

To be able to run Analysis.py you will need to **download the entire folder that contains the Analysis.py file, associated IRIS.csv file**, and picture files used in the application.

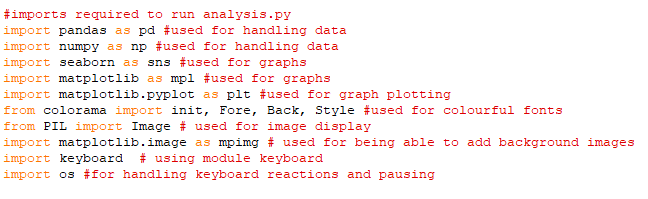
If you don’t already have python installed you will need to download version 3.7 or later from <https://www.python.org/downloads/>

While installing Python it is very important to tick the option to add Python to PATH.



**SETTING PYTHON UP TO RUN ANALYSIS.PY**

Located at the top of the source code of most python programs are a list of modules that the application uses.



Before you run **Analysis.py** PIP INSTALLS may need to get the above modules to work. In the event of errors, check the source code to see which modules need to be installed.

Generally a PIP install has to be done for any module used in the application that isn’t installed on your computer / version of Python. If the Python application isn’t loading when double clicked or run from your python editor, you will need to do PIP installs from the command prompt.

Here are examples of pip installs that you will need to do for any module not found error for this application. Get the name of module from module not found error and pip install it from your command prompt on windows of mac terminal.

**e.g. pip install colorama, pip install PIL, pip install pandas, pip install matplotlib, pip install seaborn**

|  |  |
| --- | --- |
|  | If Python has correctly been installed to PATH you should be able to initiate the PIP install command from any directory location. |

With PIP installs all done, to activate **Analysis.py** you should simply be able to double click on the file and Python will run it automatically, alternatively you can run it from Python editor of your choice such as IDLE or Visual Studio Code.

**PART 2**

THE IRIS DATASET

Linear Discriminant Analysis

Studying the Iris flower data set, is the ideal starting place to learn LDA (Linear Discriminant Analysis) and to develop data plotting and extraction skills with Python. This is because the data set is small, it contains 3 classes of 50 instances each, where each class refers to a type of iris plant. Containing only 150 observations of petal length, petal width, sepal width, sepal length the data set is small but not trivial.

Teachers lover the Iris data set because most students can relate to what flowers are, and understand the end result of being able to identity and categorise into different types. Linear Discriminant Analysis is also extensively used by geneticists in determining which genes respond well to drug tests, it is also was the first statistical method used for Bankruptcy prediction, Face recognition, and widely used in Marketing.

Rather than beginning to learn LDA through doing categorisation of a complex subject matter like genetics which could involve thousands of genes and variable factors, the Iris data set allows the beginner to be able to quickly see the result of LDA successfully with a small amount of data and a few simple commands in Python using some very powerful libraries such as Matplotlib, Pandas, Seaborn, Scikit Learn.

The Iris data set is used by students of computer programming and machine learning with the aim of LDA(Linear Discriminant Analysis), to improve computer learning processes to be able to classify iris flowers among three species (setosa, versicolor or virginica) from measurements of length and width of sepals and petals. Once understood this knowledge can then be applied to more advanced fields of computer science.

**SOURCES**

[**https://en.wikipedia.org/wiki/Iris\_flower\_data\_set**](https://en.wikipedia.org/wiki/Iris_flower_data_set)

[**https://en.wikipedia.org/wiki/Linear\_discriminant\_analysis**](https://en.wikipedia.org/wiki/Linear_discriminant_analysis)

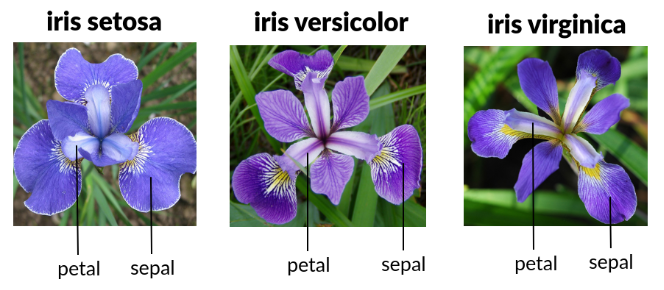


Image source: <https://www.slideshare.net/BrittanyLasseigne/an-introduction-to-machine-learning-and-genomics>

Modern day LDA technologies have roots in the work of Sir Ronald Alymer Fisher, a British statistician and geneticist, who is acclaimed for his work in statistics. In 1936 Fisher introduced the Iris flower data set as an example of discriminant analysis.

*The Iris dataset is deservedly widely used throughout statistical science, especially for illustrating various problems in statistical graphics, multivariate statistics and machine learning.* <https://stats.stackexchange.com/questions/74776/what-aspects-of-the-iris-data-set-make-it-so-successful-as-an-example-teaching>

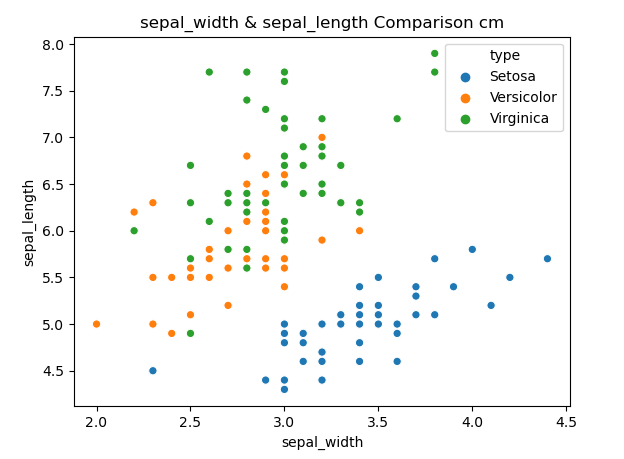
The Iris data set is such a popular data set for teaching Linear Discriminant Analysis to students of computer science and machine learning, that the data set is inbuilt and accessible in machine learning module libraries used with Python such as scikit learn and R.

**What is Linear Discriminant Analysis?** In lay persons terms, it is being able to analyse data in such a way, that ‘*When we plot the features, we can see that data is linearly separable. We can draw a line to separate the two groups. The problem is to find the line and to* ***rotate the features*** *in such a way to maximise the distance between the groups and to minimise the distance within the group.’*

*This is where Python comes extremely useful to visualising the categories of the Iris DataSet Quickly.*

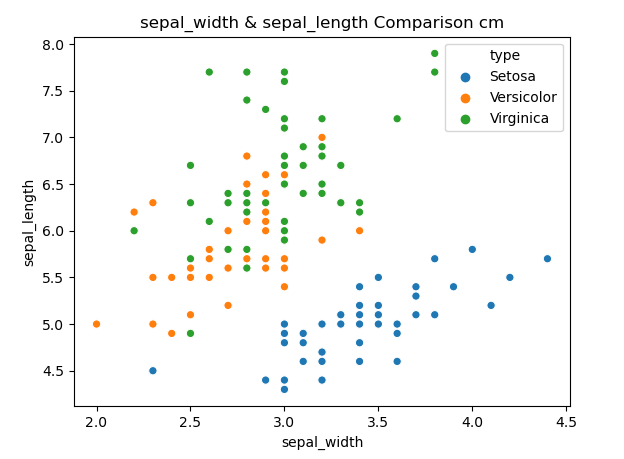
|  |  |
| --- | --- |
| *A few lines of code like this*  *import pandas as pd*  *import matplotlib.pyplot as plt*  *import seaborn as sns*  *df = pd.read\_csv("IRIS.csv")*  *sns.pairplot(df, hue="type")*  *plt.show()* | *Can generate wonderful graphs like this* |

*We will explore further this idea of rotating the features of the data set,* A cursory observation of the Fisher Iris data set using graphs plotted by Python, shows two clusters with rather obvious separation. One of the clusters (illustrated in blue) contains Iris setosa, while the other cluster contains both Iris virginica and Iris versicolor (illustrated in orange and green). Analysis of the second cluster would not be separable without the species information Fisher used.

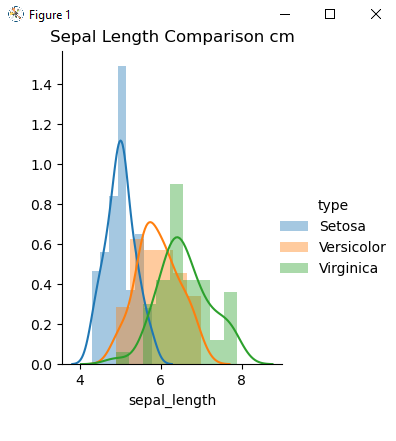
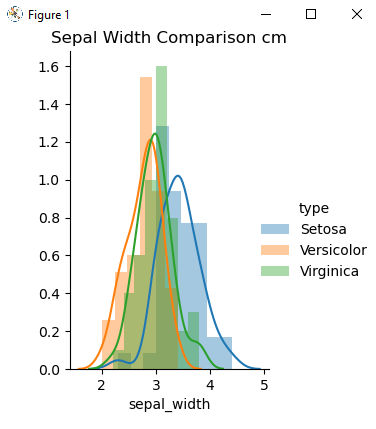
****

Only on further analysis of the proportions of sepal length, sepal width, petal length and petal width size, does it become easier to classify characteristics of the Iris Virginica and Iris Versicolor species.

Scatter graphs are ideal for doing Linear Discriminant Analysis comparing two types of data. Using the scatter plot below that knowing the relationship between the width and length of sepals, gives you an increased chance of being able to identifying either a tall Virginica or a short Setosa. However with limited data on just Sepal heights and widths it is almost impossible to categorize the difference between the average Versicolor and average Virginica, unlike the Setosa, the average data of Vericolors and Virigincas are all mixed together in a cluster.

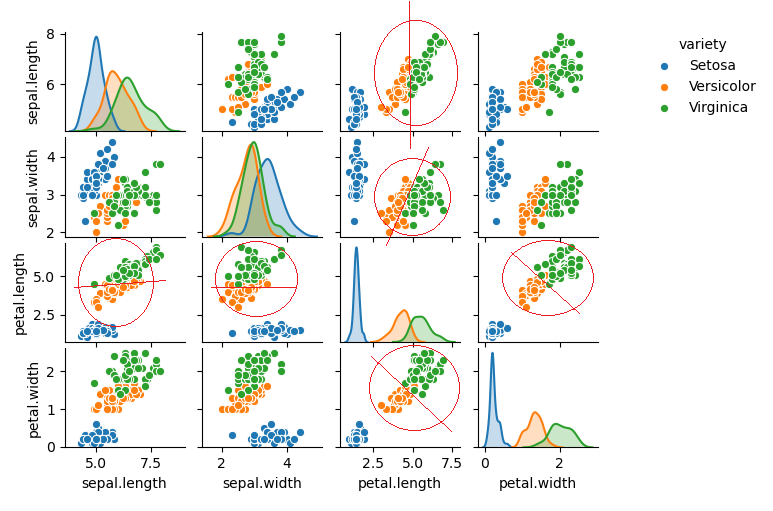
****

On limited data it is very hard to tell one plant from another simply from measurements of their sepal length or sepal widths. Histograms are used below for showing one type of data from each plant. Using these two histograms below you would be challenged to be able to tell the different between an average sized Veriscolor or Virginica from sepal sizes alone.

****

However when you **‘rotate the features’** especially with multiple scatter graphs testing multiple permutations of the available variables to see if there is a correlation between e.g sepal length and petal length or sepal width and petal length, it becomes easier to differentiate the data of the different species and maximise the distance between the groups and to minimise the distance within the group.’

Below I have edited in red the Iris data set graph from Dr Ian McLoughlin’s demonstration of Python and Seaborn, to show indicators of how measurements in the 2nd cluster emerge to distinguish one species from another.

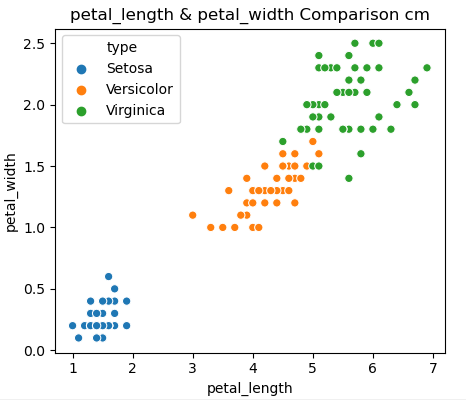


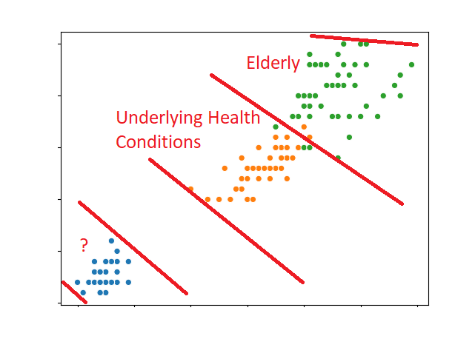
At the time of writing this project we are in the middle of a global pandemic that was originally thought to only affect the elderly and people with underlying health conditions, as the virus has spread through asymptomatic carriers the world has gone into lockdown and people of all ages with no underlying health conditions are also beginning to die of the virus.

Scatter graphs and Histograms used in Linear Discriminant Analysis are a very powerful way to visualize data and display clusters and relationships to data.

The skills we put in practicing learning the simple Iris dataset, may be the very same skills, with the right data, that will identify what are the commonalities between the mortalities of apparently healthy people and the elderly and people with underlying health conditions. With knowledge and understanding of data more effective policies and strategies can be developed in practice and in law.

Whether you are working in high security facial recognition, preventing bankruptcy of a FTSE 100 company or fighting a global pandemic. LDA and data visualisiation is cutting edge, and you can become experience in it using freely available Python and powerful libraries such as Matplotlib, Pandas, Seaborn, Scikit Learn.

**Today Sepals & Petals For Fun Tomorrow Prevention of Covid19**

****