Linear Discriminant Analysis

**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.**

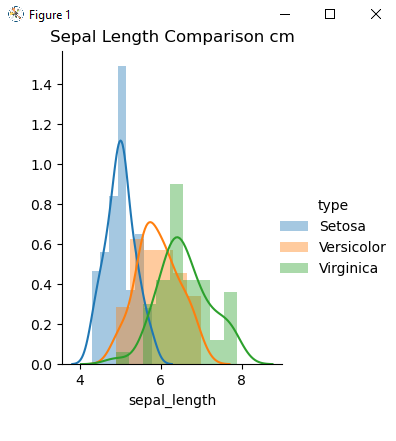
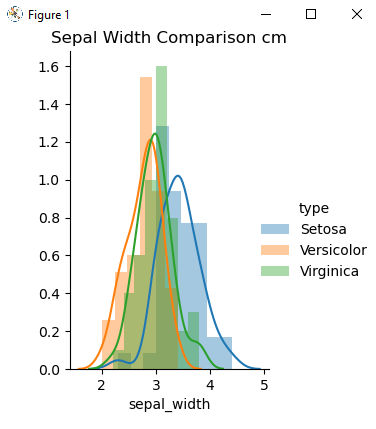
**Scatter graphs used in Linear Discriminant Analysis are a very powerful way to display clusters and relationships to data.**

**One day, the skills put in practice learning the Iris Dataset, may be the very same skills that save lives from the global pandemic.**

**The graph below is a scatter plot graph generated by the seaborn module of Python. Its an extremely simple way to see at a glance that Setosa Iris plants have a short petal length and petal width. Versicolor and Virginic plants cluster together in a second group**

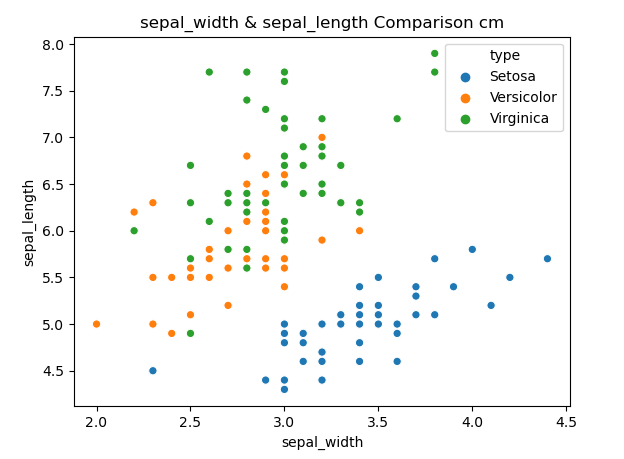
**THE IRIS DATASET**

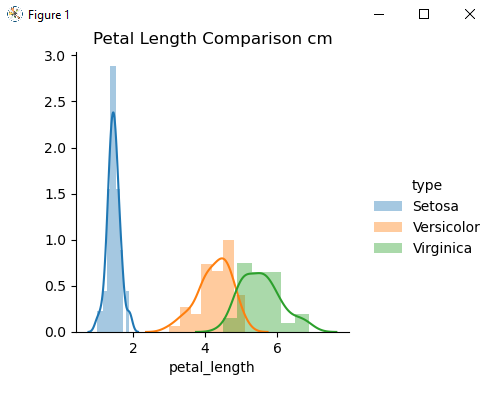
**On limited data it is very hard to tell one plant from another simply from measurements of their sepal length or sepal widths. Histograms are for showing one type of data from each plant.**

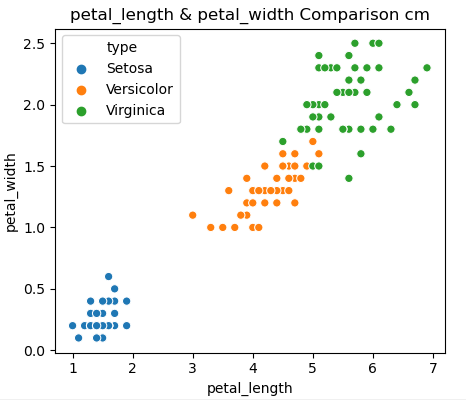
****

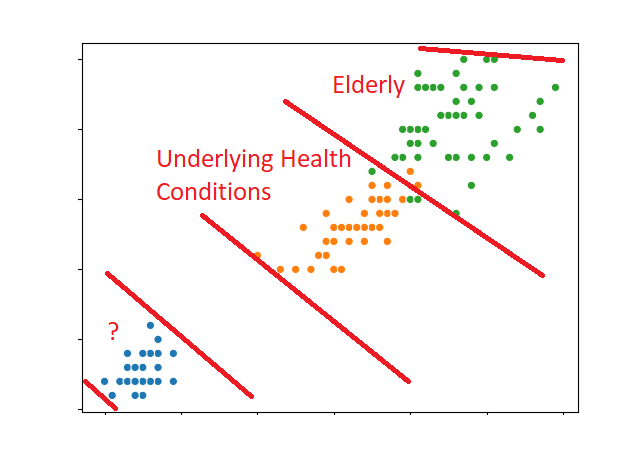
**Scatter graphs are ideal for comparing two types of data.**

**Already we can see from 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.**

****

****

****

****

**CONTENTS**

**Iris Data Set**

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

**MENU**

**====**

**1 – Introduction To Iris Data Set**

**2 - View Image Of Iris Varieties**

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

**3 – View Average Sizes Iris**

**4 – View Minimum Sizes Iris**

**5 – View Maximum Sizes Iris**

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

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

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

**7 – View Paired Graph Plots")**

**8 – View Scatter Plots**

**9 – View Histograms**

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

**10 – Save Paired Graph Plots**

**11 – Save Scatter Plots**

**12 – Save Histograms**

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

**x – Exit application**

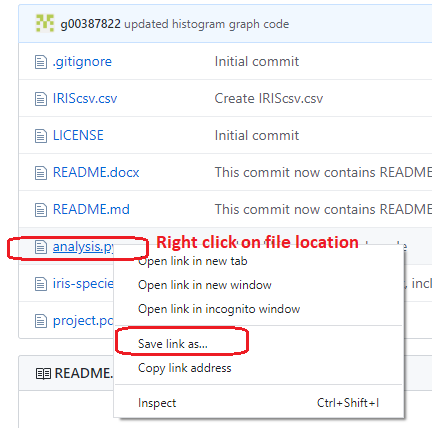
**PYTHON APPLICATION SUMMARY**

Over the next few pages I shall share screenshots of the twelve menu sections of a simple Python program I designed called Analysis, which can be used to learn Python commands that I have 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 to some of the basic commands and modules that are available for use with Python that can assist further analysis.

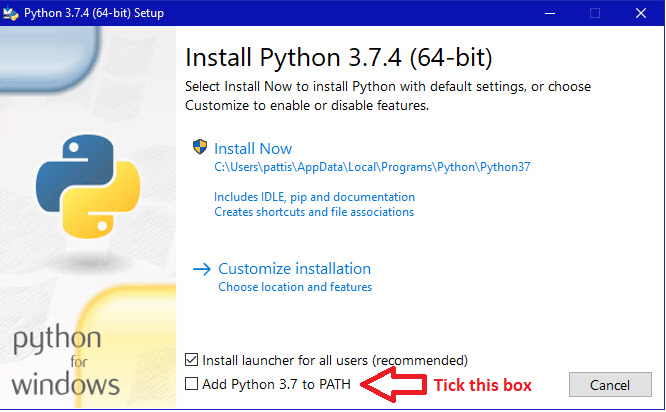
To be able to run Analysis.py you will need to **download the Analysis.py file and the IRIScsv.csv file** from my github <https://github.com/g00387822/pands-project>



Both the Analysis.py file and the IRIScsv.csv file will need to be saved in the same directory.

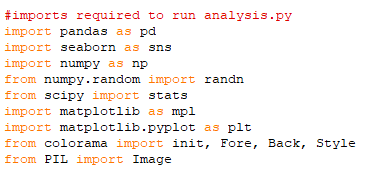
If you don’t already have python installed you will need to download version 3.7 or later from <https://www.python.org/downloads/>

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.

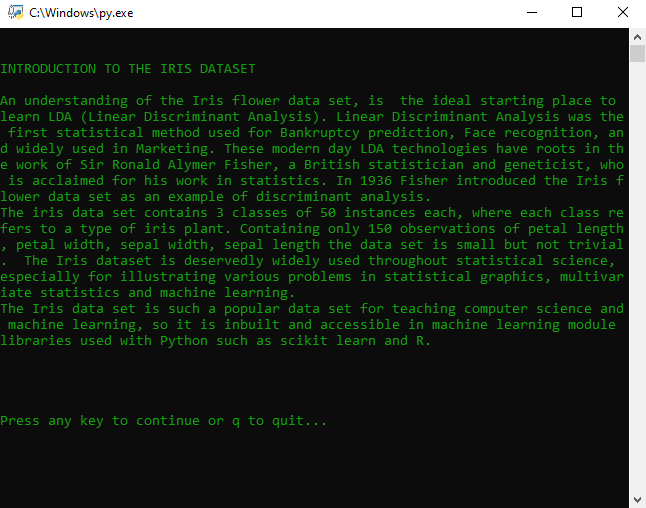
**e.g. pip install colorama, pip install PIL, pip install pandas, pip install matplotlib, pip install seaborn**

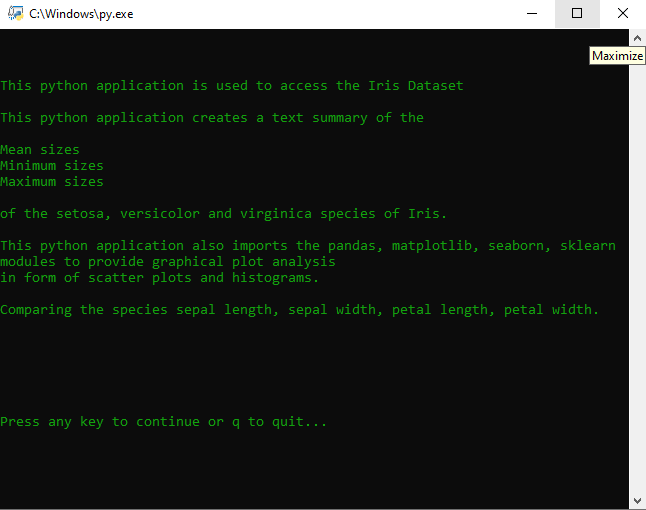
|  |  |
| --- | --- |
|  | If Python has correctly been installed to PATH you should be able to initiate the PIP install command from any directory location. |

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.

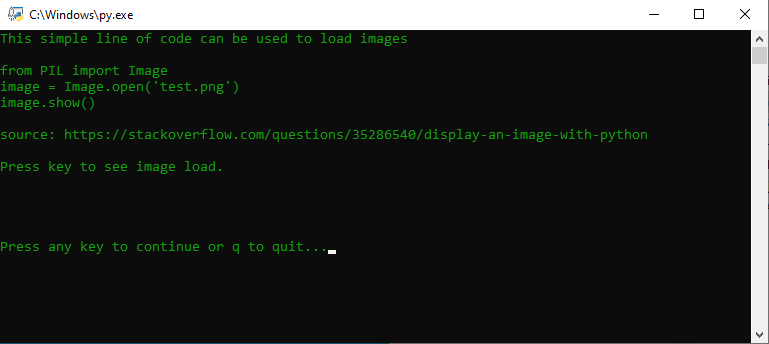
**MENU 1. Introduction To Iris Data Set.**

**This section of Analysis.py is just a simple description about the Iris Dataset.**

****

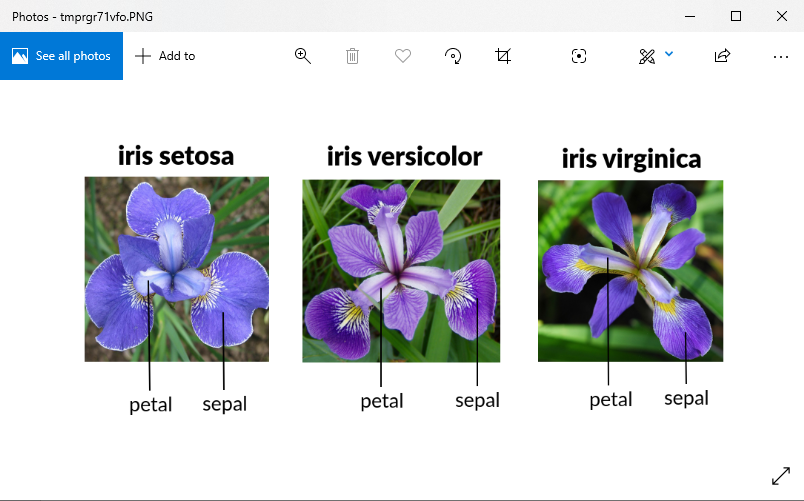
**It is followed by a very simple description about the python application.**

**MENU 2. View Image Of Iris Varieties**



A picture says a thousand words. One of the easiest ways to convey to an audience the subject matter of what you are trying to convey is share an image. My python application provides the code of how to access an image using Python, and on the next key press shows the image.

|  |
| --- |
| from PIL import Image  image = Image.open('test.png')  image.show()  source: https://stackoverflow.com/questions/35286540/display-an-image-with-python |

****

**MENU 3. View Average Sizes Iris; MENU 4. View Minimum Sizes Iris**

**MENU 5. View Maximum Sizes Iris**

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 are coded to get results straight from the CSV file. These are demonstrated in menu items 3,4,5.

|  |  |
| --- | --- |
|  | # importing pandas as pd  import pandas as pd  # Creating the dataframe  IrisData = pd.read\_csv('IRIScsv.csv')  # sum over the column axis.  averageofdata = IrisData.mean(axis = 0, skipna = True)  print("Average Sizes of All Iris Data")  print(averageofdata) |

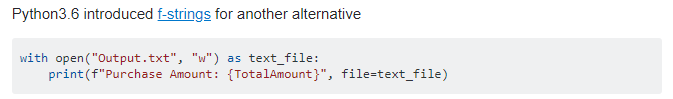
Sources: <https://cmdlinetips.com/2018/02/how-to-subset-pandas-dataframe-based-on-values-of-a-column/>

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

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

**MENU 6. Save Summary Data To Text File**

As requested in the project specification I have implemented a method to save the variable results from Python / Panda Dataframes to a text file.



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

|  |
| --- |
| def Save\_Summary\_Of\_Average\_Iris\_Sizes\_To\_Text\_File():  print("Save Summary Of Average Iris Sizes To Text File")  myfile = input("Write a file name ending with .txt , this is where average, minimum, maximum data summarys will be saved ...")  fileforappending = open(myfile, "w")  print(myfile + " has been created")  fileforappending.close()  print(myfile + " has been created and is ready for appending data.")  print("You will find your file here: ", os.getcwd(), "\\" , myfile)  print("Average Of All Iris", file=open(myfile, 'a'))  print(**pd.DataFrame**(averageofdata), file=open(myfile, 'a'))  print("Average Of Setosa", file=open(myfile, 'a'))  print(pd.DataFrame(average\_of\_Setosa), file=open(myfile, 'a'))  print("Average Of Versicolor", file=open(myfile, 'a'))  print(pd.DataFrame(average\_of\_Versicolor), file=open(myfile, 'a'))  print("Average Of Virginica", file=open(myfile, 'a'))  print(pd.DataFrame(average\_of\_Virginica), file=open(myfile, 'a'))  print("Minimum Of All Iris", file=open(myfile, 'a'))  print(pd.DataFrame(minimum\_of\_data), file=open(myfile, 'a'))  print("Minimum Of Setosa", file=open(myfile, 'a'))  print(pd.DataFrame(minimum\_of\_Setosa), file=open(myfile, 'a'))  print("Minimum Of Versicolor", file=open(myfile, 'a'))  print(pd.DataFrame(minimum\_of\_Versicolor), file=open(myfile, 'a'))  print("Minimum Of Virginica", file=open(myfile, 'a'))  print(pd.DataFrame(minimum\_of\_Virginica), file=open(myfile, 'a'))  print("Maximum Of All Iris", file=open(myfile, 'a'))  print(pd.DataFrame(maximum\_of\_data), file=open(myfile, 'a'))  print("Maximum Of All Setosa", file=open(myfile, 'a'))  print(pd.DataFrame(maximum\_of\_Setosa), file=open(myfile, 'a'))  print("Maximum Of All Versicolor", file=open(myfile, 'a'))  print(pd.DataFrame(maximum\_of\_Versicolor), file=open(myfile, 'a'))  print("Maximum Of All Virginica", file=open(myfile, 'a'))  print(pd.DataFrame(maximum\_of\_Virginica), file=open(myfile, 'a')) |

**MENU 7. View Data As Paired Graph Plots**

I found that the easiest and most visual way for me to generate the best results to demonstrate the Iris data set was following the method demonstrated by Dr Ian McLoughlin using Python and Seaborn.

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

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

|  |  |
| --- | --- |
|  | **import pandas as pd**  **import matplotlib.pyplot as plt**  **import seaborn as sns**  **df = pd.read\_csv("IRIScsv.csv")**  **sns.pairplot(df, hue="variety")**  **plt.show()** |
|  | |

**MENU 8. View Data As Scatter Plot**

It is very easy to generate individual Scatter Plots using

|  |
| --- |
|  |
| import pandas as pd  import matplotlib.pyplot as plt  def View\_Data\_As\_Scatter\_Plot():  print("Save Scatter Plot")  df = pd.read\_csv("IRIScsv.csv")  plt.scatter(df['sepal.length'], df['sepal.width'])  plt.title("Sepal length versus sepal width")  plt.xlabel("Sepal Length")  plt.ylabel("Sepal Width")  plt.show()  plt.scatter(df['petal.length'], df['petal.width'])  plt.title("Petal length versus sepal width")  plt.xlabel("Petal Length")  plt.ylabel("Petal Width")  plt.show() |

**MENU 9. View Data As Histogram**

2. Download the data set and add it to your repository.

<https://github.com/g00387822/pands-project/blob/master/IRIScsv.csv>

3. Write a program called analysis.py

<https://github.com/g00387822/pands-project/blob/master/analysis.py>

• outputs a summary of each variable to a single text ﬁle

• saves a histogram of each variable to png ﬁles,

and • outputs a scatter plot of each pair of variables.

**IRIS DATA SET SUMMARY**

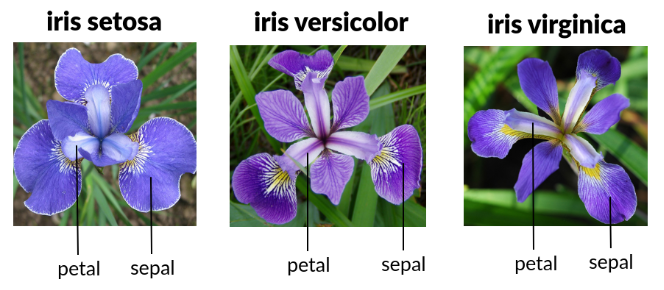


Image source: <https://www.slideshare.net/BrittanyLasseigne/an-introduction-to-machine-learning-and-genomics>

An understanding of the Iris flower data set, is the ideal starting place to learn LDA (Linear Discriminant Analysis). Linear Discriminant Analysis was the first statistical method used for Bankruptcy prediction, Face recognition, and widely used in Marketing. These 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.

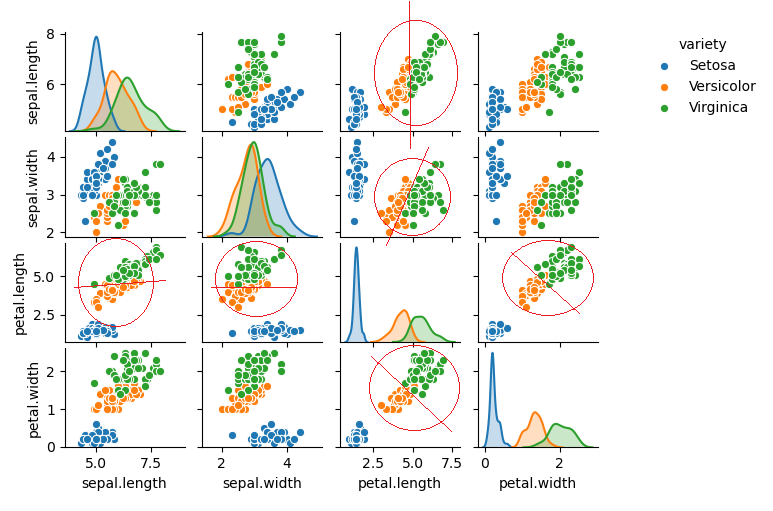
The iris data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. Containing only 150 observations of petal length, petal width, sepal width, sepal length the data set is small but not trivial. *The Iris dataset is deservedly widely used throughout statistical science, especially for illustrating various problems in statistical graphics, multivariate statistics and machine learning.*

The Iris data set is such a popular data set for teaching computer science and machine learning, so it is inbuilt and accessible in machine learning module libraries used with Python such as scikit learn and R.

A cursory observation of the data set, 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 where 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.

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.



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. This understanding can then be applied to more advanced fields of computer science.