Machine Learning Using Scientific Python

Analysis of Fisher's Iris Dataset

Supervised Project

Bachelor of Computer Science (through Erasmus +)

University of Alcala, Madrid

by Mariam Zenaishvili

June 2022

Contents

Housekeeping **2**

Python Files in this Repository 2

How to Download this Repository 2

What is Needed to Run the Python Scripts 2

Introduction **2**

Background 3

Why is the Iris Dataset Still Relevant? 3

Analysis of the Dataset **4**

Conceptualizing the Dataset 4

Plotting the Dataset 6

Histograms 6

Box Plots 8

Scatter Plots 10

Parallel Coordinates 13

Finding Correlations Between the Variables 14

Using Linear Regression 15

Statistical Classification of the Data 25

Linear Discriminant Analysis of the Dataset 25

Conclusion **33**

Appendix **33**

Definition of Key Terms 33

References **36**

Abstract

The purpose of this project is to investigate "The Iris Flower Data Set" to prove my understanding of the data set and explain its importance to machine learning. This project is being undertaken as a part of a final submission for the subject 'Programming and Scripting' as a part of my Diploma in Computer Science with Data Analytics. I intend to investigate, research, and produce my findings via this submission. My goal for this project is to provide education within the field of machine learning so as to provide knowledge to those interested in understanding it in more detail. This project has been approached with the intention of researching all information of the Iris Flower Data Set, writing and running scripts to test the set within Python, and then providing written support of my view and knowledge of the data set.

Housekeeping

Python Files in this Repository

- an 'analysis' file which is the main application script and which imports and runs the functions in the other files
- a 'createBoxplots' file that saves boxplots overlayed with swarm plots to the plots\boxplots folder
- a 'createDataFrame' file that creates a pandas dataframe object based on the irisDataSet file
- a 'createHistograms' file that saves histograms for to the plots\histograms folder
- a 'createParallelCoordinates' file that saves a parallel coordinates plot to the plots\parallelCoordinates folder
- a 'createScatterPlots' file that saves scatter plots to the plots\scatterPlots folder
- a 'createSummaryStatistics' file that saves summary statistics for each species to a csv file in the summaryStatistics folder
- a 'dimensionalityReduction' file that performs various dimensionality reductions and saves the plotted results to files in the plots\dimensionalityReduction folder
- a 'verifyIntegrityOfDataset' file to do just that

How to Download this Repository

Click on the green 'Clone or download' button near the top right of this page, and then select 'Download ZIP'. You will then need to unzip the downloaded file.

What is Needed to Run the Python Scripts

Python 3.7.4 was used to create the scripts in this repository. Any version of Python 3 can run the scripts. You can also download Python by downloading the Python distribution, Anaconda.

Introduction

This report's analysis of the Iris Dataset is structured as follows:

- The dataset is described
- The data is conceptualized and questions about the dataset are formulated based on this conceptualization
- The data is plotted, and rudimentary visual analyses of the data are undertaken based on the plots
- The relationships between the variables in the dataset are analyzed with linear regression techniques
- Statistical classification of the data is undertaken using a supervised learning technique (linear discriminate analysis)

Background

The Iris flower data set or Fisher's Iris data set is a multivariate (these blue links will take you to a definition in the appendix section below) data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper *The use of multiple measurements in taxonomic problems as an example of linear discriminate analysis. The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica, and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Based on the combination of these four features, Fisher developed a linear discriminant model to distinguish the species from each other. The Iris Flower Data Set itself is extremely popular and utilized as a testing tool within machine learning. Part of its popularity I believe, is due to its reliability and ease of use. One contributor on Stackflow described the reasons for its high accuracy: "(a) there are few features to begin with and (b) that there are high linear correlations with class, would all point to a less complex, linear function as being the appropriate predictive model-- by using a single hidden node, you are very nearly using a linear model. It can also be noted that, in the absence of any hidden layer (i.e., just input and output nodes), and when the logistic transfer function is used, this is equivalent to logistic regression." [7] While another contributor responded to the same post with a sample test for the data set:

" The Iris data set can even be predicted with a very high accuracy (96%) by using just three simple rules on only one attribute: If Petal.Width = (0.0976,0.791] then Species = setosa If Petal.Width = (0.791,1.63] then Species = versicolor If Petal.Width = (1.63,2.5] then Species = virginica "
[7]

Why is the Iris Dataset Still Relevant?

The Iris Dataset remains relevant because:

- Its analysis by Fisher was a foundational moment in the history of data analytics, particularly in relation to the development of linear discriminate analysis.
- It is well suited to be used as an educational tool in data analysis: the sample size is small at 150 flowers; the data is of that of everyday life (a common flower rather than some deep-sea flora); the variables are not intimidatingly scientific (e.g. petal length); and the measurements are easily imaginable (a few centimeters).
- It can be used to explain the basic difference between supervised learning and unsupervised learning techniques in machine learning.
- Consisting of three species of the Iris flower, two of which are not immediately distinguishable based on the measurements obtained, the Iris dataset has often been used to test statistical classification techniques in machine learning, i.e. how to teach a program to categorize an object based on its attributes.

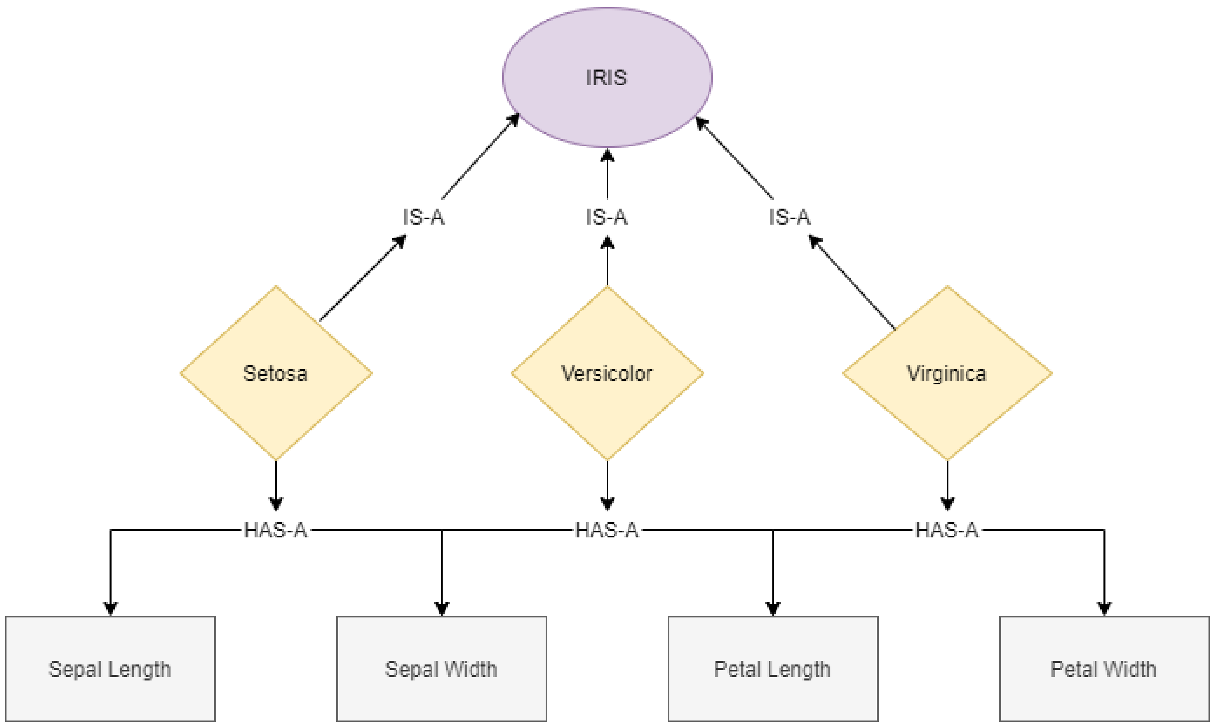
Analysis of the Dataset

Conceptualizing the Dataset

While it's all very well to say that the dataset consists of three species of iris flowers and their petal and sepal lengths and widths, it is also important to have an understanding of what the data, for lack of a better word, means. This can be achieved by conceptualizing the relationships between the variables in the dataset; and doing this will bring one a good part of the way towards understanding what the dataset can actually be used for.

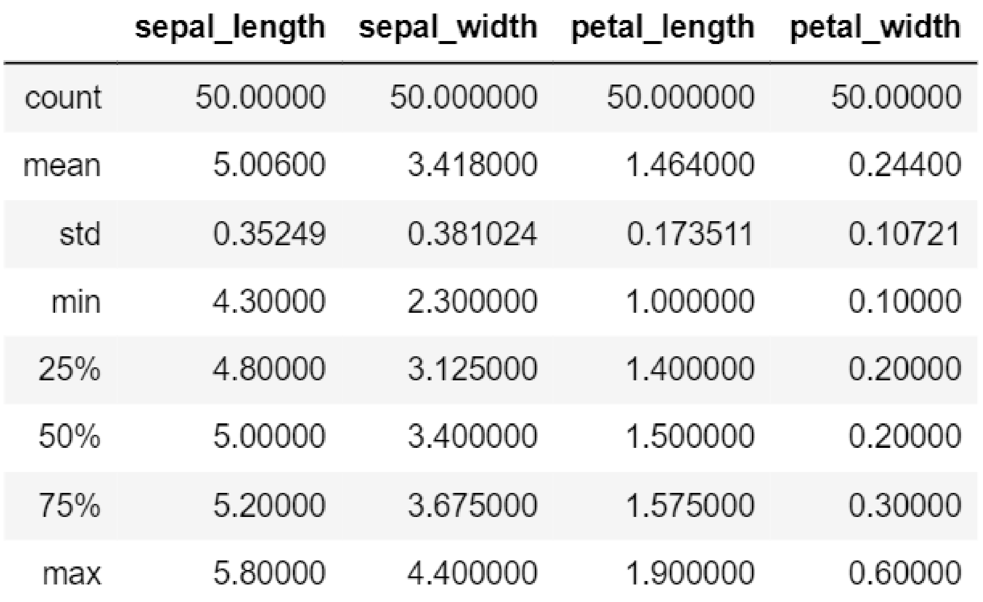
In a sense we have done a certain amount of conceptualizing already: although the three species of flower are represented as nothing more than variables in the dataset - no different from petal length, for example - we have already been referring to them as something more than just a variable, as a special kind of variable, as it were.

The differentiation made in Object Oriented Programming between 'Is-a' and 'Has-a' relationships effectively captures the difference between the species variable and the other variables in the dataset: each species is an iris flower while each species has a sepal width, for example.



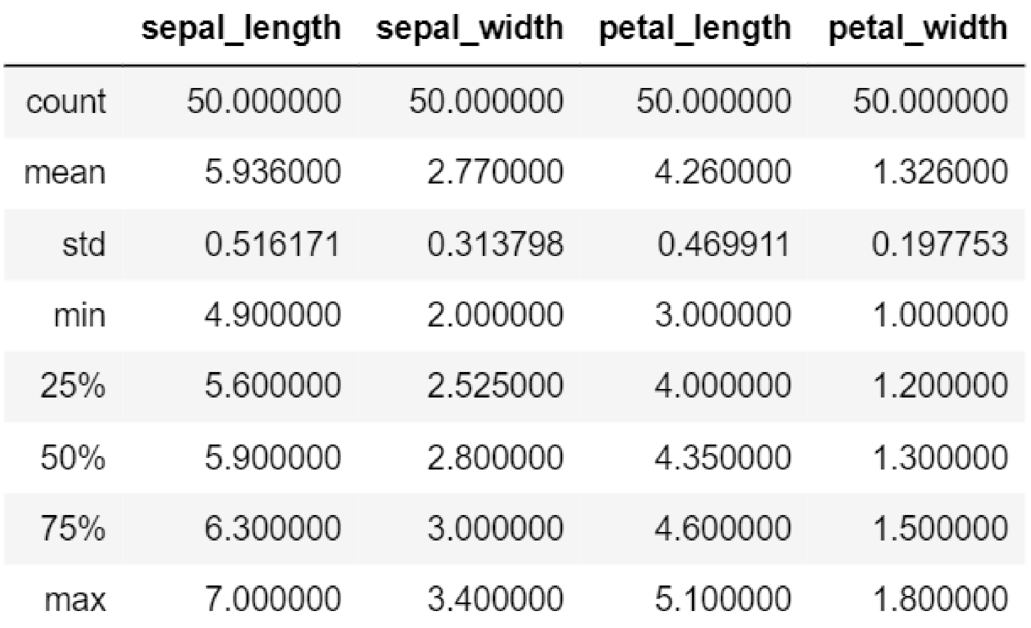
What does this mean then for our analysis of the dataset? Well, we could probably guess that it would be interesting to compare the species against each other, and indeed that this would likely be much more interesting than looking at the dataset as a whole without distinguishing between the species. Already the 'anchor' question that any analysis of the dataset is likely to confront arises: can the species be distinguished purely based on their respective sepal and petal lengths and widths? And this is exactly where the machine learning analyses of the dataset begin: could a program be taught to determine the species of an iris flower based on its sepal and petal length and width? But we are jumping the gun here, and while it is important to come into any dataset primed with questions, there is only so far one can get without getting one's hands dirty with the data.

Setosa Summary Statistics



	sepal_length	sepal_width	petal_length	petal_width
count	50.00000	50.000000	50.000000	50.00000
mean	5.00600	3.418000	1.464000	0.24400
std	0.35249	0.381024	0.173511	0.10721
min	4.30000	2.300000	1.000000	0.10000
25%	4.80000	3.125000	1.400000	0.20000
50%	5.00000	3.400000	1.500000	0.20000
75%	5.20000	3.675000	1.575000	0.30000
max	5.80000	4.400000	1.900000	0.60000

Versicolor Summary Statistics



	sepal_length	sepal_width	petal_length	petal_width
count	50.000000	50.000000	50.000000	50.000000
mean	5.936000	2.770000	4.260000	1.326000
std	0.516171	0.313798	0.469911	0.197753
min	4.900000	2.000000	3.000000	1.000000
25%	5.600000	2.525000	4.000000	1.200000
50%	5.900000	2.800000	4.350000	1.300000
75%	6.300000	3.000000	4.600000	1.500000
max	7.000000	3.400000	5.100000	1.800000

Virginica Summary Statistics

	sepal_length	sepal_width	petal_length	petal_width
count	50.00000	50.000000	50.000000	50.00000
mean	6.58800	2.974000	5.552000	2.02600
std	0.63588	0.322497	0.551895	0.27465
min	4.90000	2.200000	4.500000	1.40000
25%	6.22500	2.800000	5.100000	1.80000
50%	6.50000	3.000000	5.550000	2.00000
75%	6.90000	3.175000	5.875000	2.30000
max	7.90000	3.800000	6.900000	2.50000

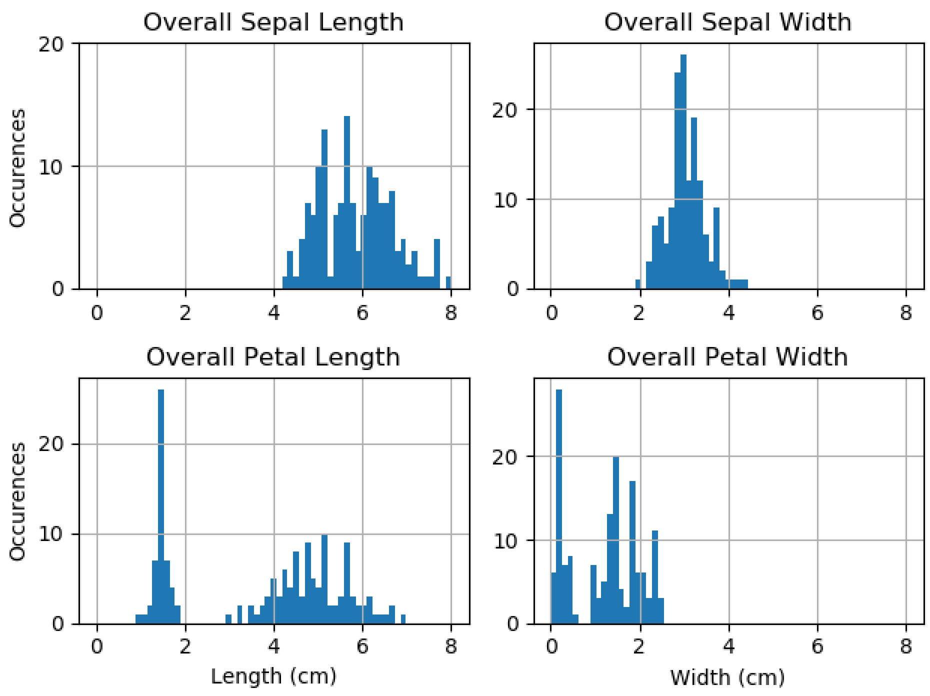
Now wasn't that just supremely uninteresting. I'm afraid these kind of statistics simply don't lend themselves to interesting analysis, although the means and standard deviations are of course important values that will be performing more detailed analyses such as regression analysis.

Plotting the Dataset

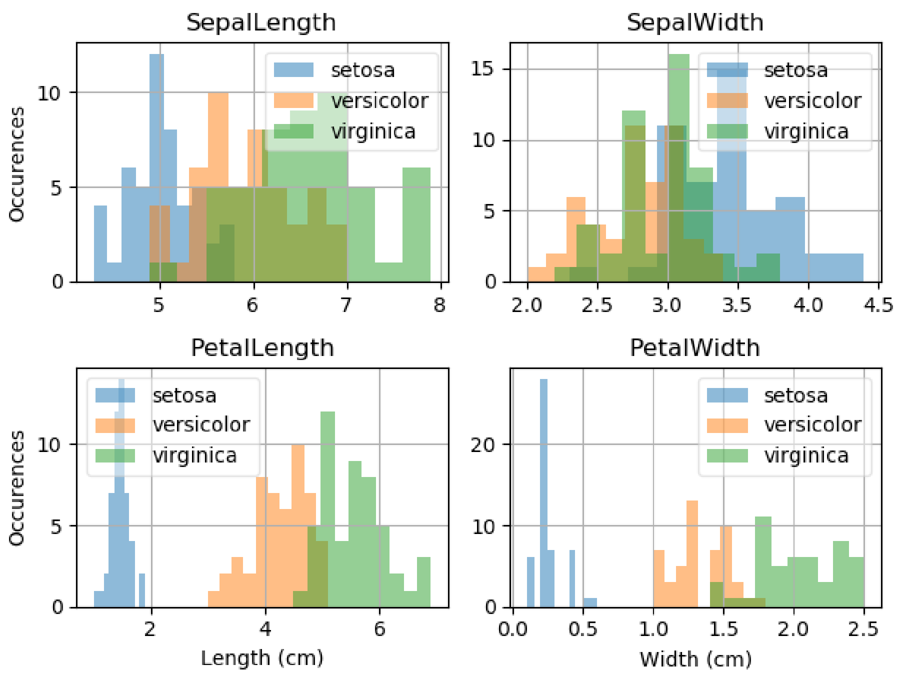
Histograms

o plotting histograms doesn't exactly constitute getting one's hands dirty: histograms are by nature heavily sanitised. They are easy to read, but don't reveal much about anything. And yet histograms remain a nice entry point into any dataset; they give one a rough idea of how the values for each variable are spread out or clumped together. Because histograms are easy to read, there is more value in including four of them in the one image rather than separately - this way one can at least see how the spread of the variables compare against each other. I have included two groups of histograms here, one with species undistinguished, and one with them distinguished.

In the case of the first group (histograms for each of the variables independent of species), I decided not to vary the bins across the histograms. While this is not ideal in the sense that there will be a lot of "white space" on some of the histograms, I think this is made up for by the ease with which one can compare the histograms, i.e. one does not have to factor in the range of the bins but can compare the variables and species purely by looking at the plots. Eight is the upper bound (virginica sepal length) and zero is the lower (setosa petal width). While bins of 0.25 width are probably the most visually appealing, the setosa-petal values are clumped together so much that bins with a width of 0.25 really aren''t granular enough. Bins of 0.125 have been used instead.



In the case of the second group (histograms for each of the variables where the species are distinguished), because we are concerned here with comparing the species against each other on the same axes, there is less need to have the x-ticks, y-ticks and bins the same across all axes; we are not so much comparing the different plots as the comparing the species within each of the plots. As such we can optimize the scale and ticks for each of the plots, and simply allocate 10 bins for each plot.

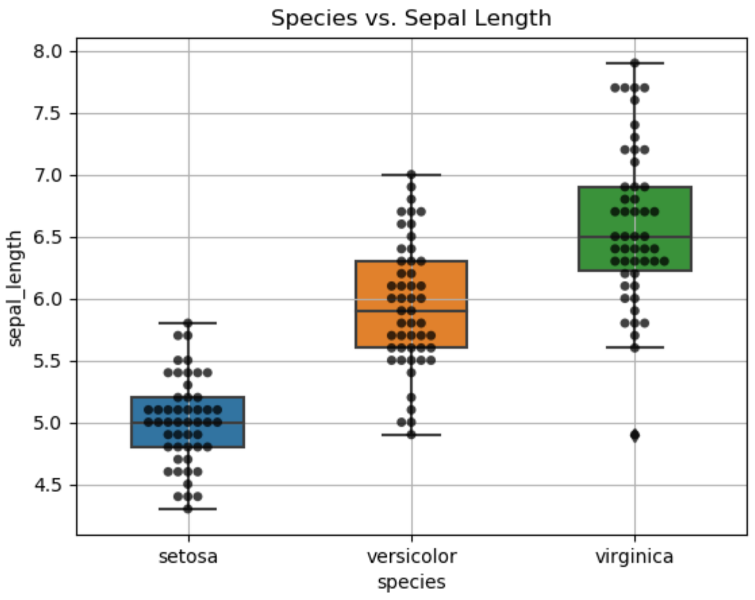


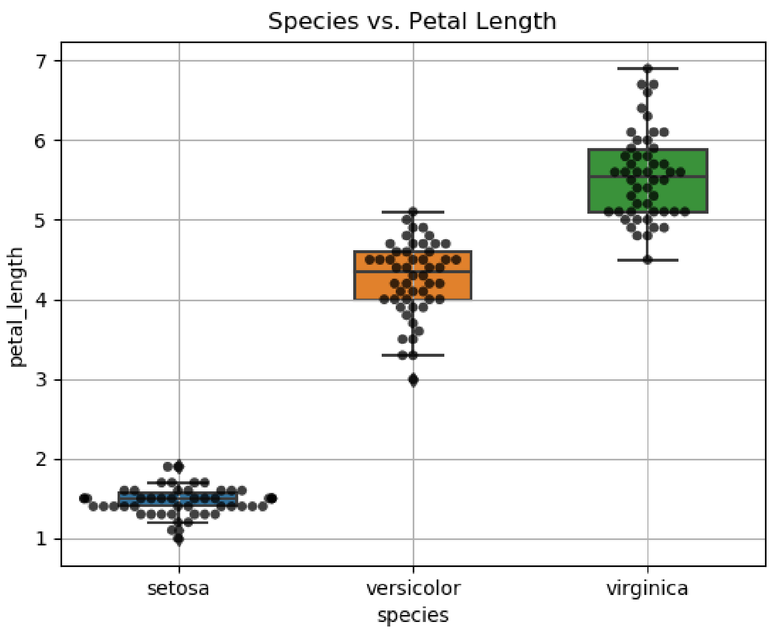
Clearly, the second group of histograms are much more interesting. The first group only show us that the variables have different ranges of values; but this difference is of course to be expected. The plots in the second group on the other hand can at least show us where there are differences between the species themselves. We can see that the species differ much in their respective petal lengths and widths, whereas there is much overlap in terms of their sepal lengths and widths. We can also see that the setosas are much more distinguishable than the versicolors and virginicas, which both have overlapping ranges of variable values. Indeed, based on the histogram for petal lengths, we could create a function to determine whether or not an iris flower was a setosa based on its petal length: if its petal length is less than 2.5 cm, then it is a setosa, if greater then it must be either a versicolor or a virginica. But once we have sucked dry that particularly clean division between the species, the histograms stop talking to us. Other plots will have to be probed.

Box Plots

Box plots are perhaps the natural progression from histograms. Like histograms they show the spread of the values of each variable; unlike histograms, however, they only distinguish between the quartiles. While they thus give us a less granular perspective than histograms, it can often be particularly interesting to see how the ranges of values in each quartile differ. Generally one would expect the ranges of values in the first and fourth quartiles to be greater than those of the second third, as is the case in a normal distribution. Box plots can be interesting then if they should that a variable does not have a normal distribution. As it turns out, the variables do have normal distributions, so to make these box plots more bearable, I have superimposed swarm plots onto them, which offer the granularity that box plots lack.

I have chosen to show the box plots for sepal and petal lengths - with the species distinguished - because at least these are somewhat interesting to compare: the ranges of values for sepal length is far greater than for petal length, and the narrow range of values for setosa petal length contrasts with all the other ranges.

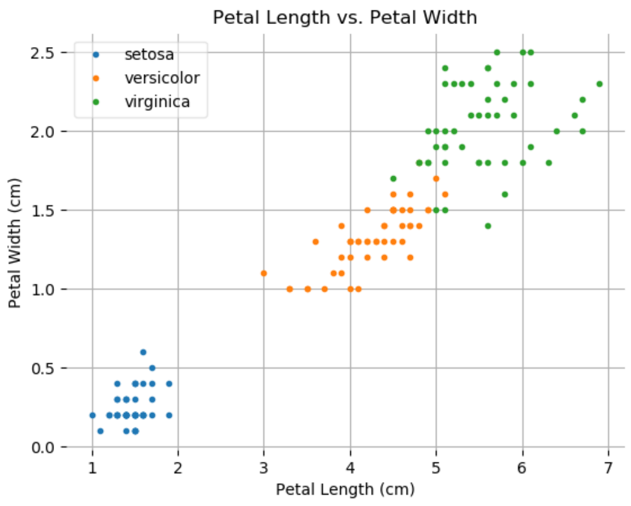




However, these box plots only tells us what the histograms have already told us: that of the four variables in the dataset petal length is likely to be the best discriminant for the species of an iris flower. As Jay-Z once said: 'on to the next one'.

Scatter Plots

Finally, we can actually begin to touch the individual data points and get to grips with the grime of the irises. Scatter plots allow us to move on from looking at variables in isolation to looking at the relationships between them. While histograms and box plots can tell us if the species' variables have differing or similar ranges of values, they cannot tell us if there is any relationship between the variables - whether or not, for example, irises with long petals also tend to have wide petals. And with that we have our first scatter plot, and perhaps the most visually appealing plot of the iris dataset:

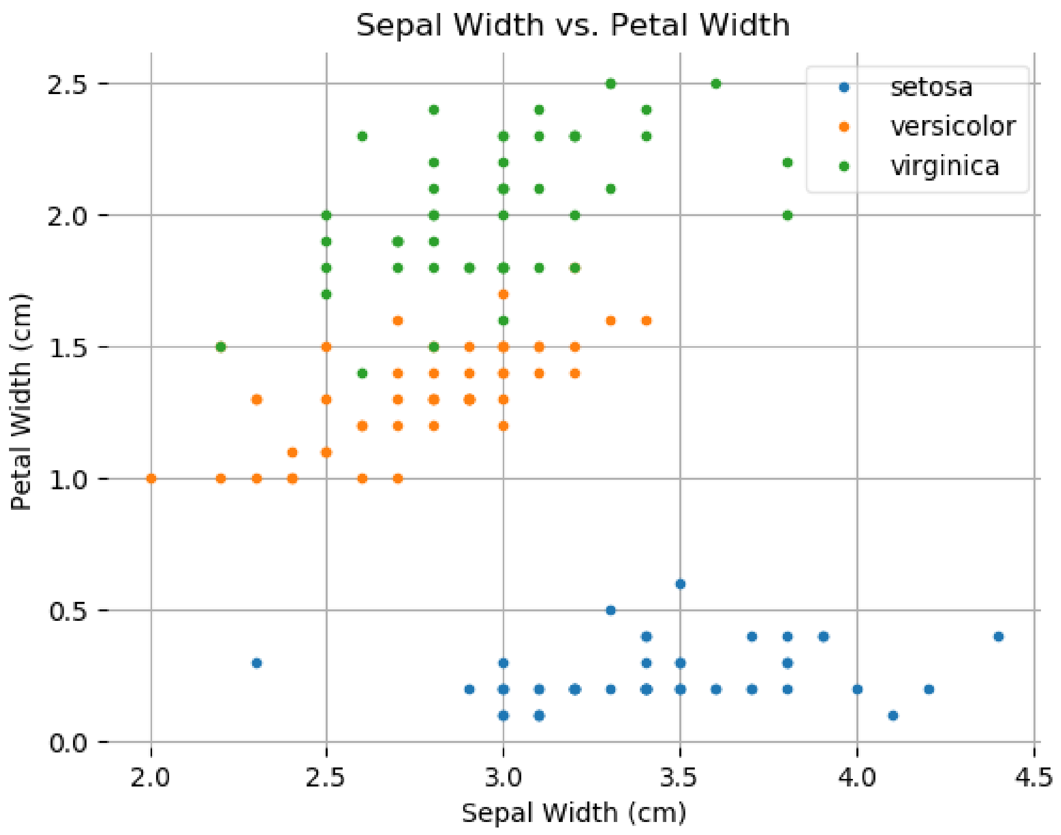


Mariam Zenaishvili | June, 2022

Of course, while such a scatter plot is satisfying to look at, in a certain sense it is not very interesting. Clearly, for all three species, as petal length increases so too does petal width. It is the clearly here that makes the plot not all that interesting. Indeed, the species are even clearly distinguished in this plot, with only a small overlap between versicolors with long and wide petals and virginicas with short and thin petals. What is interesting is that even though the histogram for petal lengths already showed us that petal length was able to determine if an iris was a setosa or not, it did not clearly suggest that it could also be used with reasonable accuracy to also determine whether an iris flower is a versicolor or virginica. This is because in the histogram the data is not represented directly; rather we are seeing the bins, and thus there appears to be much more overlap than their actually is.

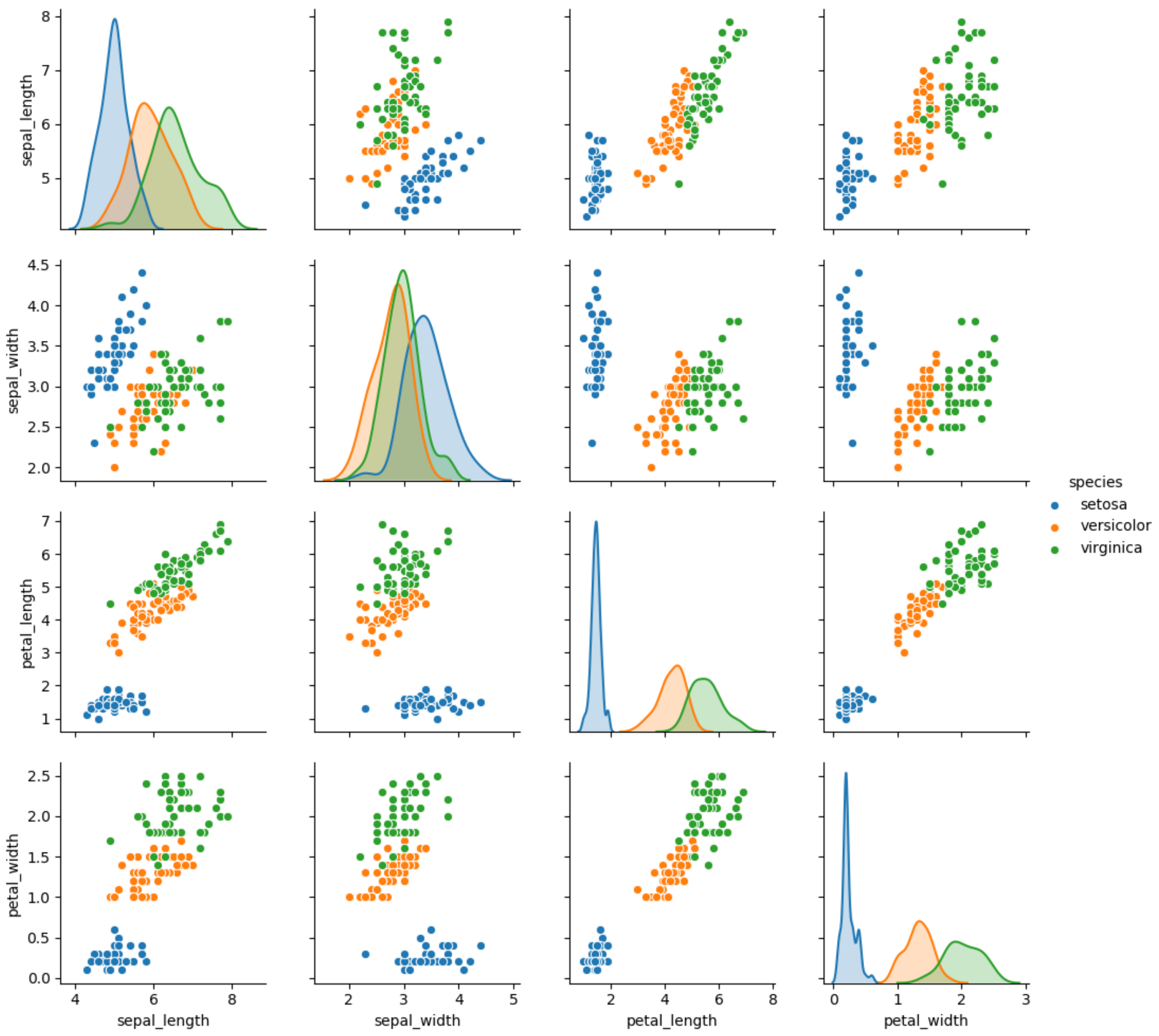
Exactly because the above plot is so easy to read and provides us with interesting, it is the kind of plot that a data analyst should in many ways dread; it is so easy to interpret that it does away with the need for the kind of sophisticated analysis that supposedly only data analysts can provide...

We should look at a more ambiguous plot, then - enter sepal width versus petal width (you can hear Jay-Z's encouragemant I hope):



so there are clearly still some trends here. Versicolors' and virginicas' sepal widths visibly tend to increase with their petal widths, but other than that? There doesn't appear to be any relationship between setosa sepal and petal width, and the versicolors and virginicas appear to be indistinguishable in terms of their sepal widths. But this is just where our first interesting questions appears: will it be at all possible to distinguish the versicolors and the virginicas based on their sepal and petal lengths and widths?

Apart from asking this question and querying the plots with are eyes to hedge an answer, and apart from finding such clean correlations as in the plot of petal length versus petal width, the scatter plots can't reveal much more to us. The next step then would be to look at each of the possible scatter plots and see how distinguishable the species are in them as well as if the variables are correlated with each other. Hence, the we arrive at the infamous pair plot / scatter matrix:

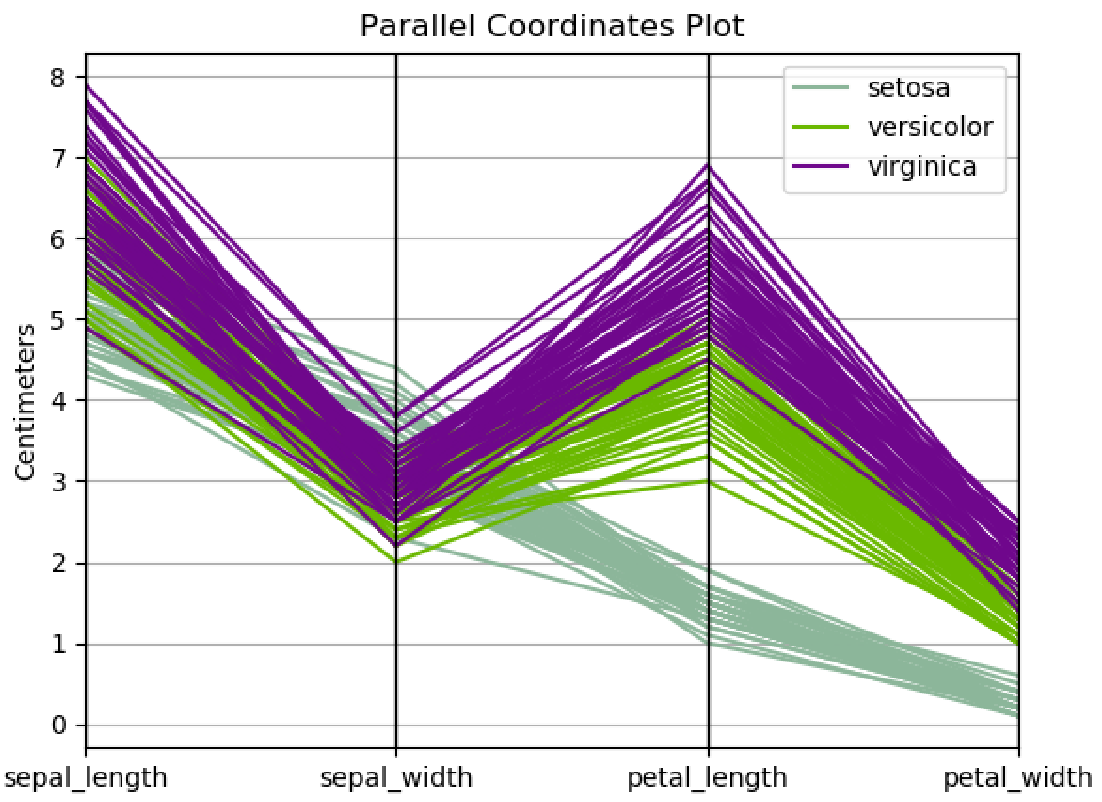


Looking at all of the scatter plots in such a matrix makes it very easy to see at a glance the relationships between the species and between the sepal and petal lengths and widths of each of the species. We can see that in many cases there doesn't appear to be a strong correlation between the lengths and widths, while in other cases it appears there are. Likewise we can very clearly see that the setosas seem to live in a world of their own. What is perhaps most intriguing is how the virginicas almost appear as extensions of the versicolors: in many of the plots their variables appear to correlated in the same way with the only difference being that the virginicas' variables' values are greater - if one didn't know that they were indeed different species, one would be forgiven for assuming that they were in fact the same species, with the virginicas accounting for the third and fourth quartiles of values and the versicolor the first and second. But again, while this is all very interesting, there is only so much one can do with the plots. In effect even the scatter plots are not that grimy after all; in order to really get into the muck of the dataset one has to engage with the false gods of mathematics.

Parallel Coordinates

Before taking the dive into the realm of unforgiving functions, there is one last plot that is worth having a look at.

The parallel coordinates plot is hands down, no question, absolutement, god-forgive-you-if-you-doubt-it the most beautiful of plots of the iris data set. I mean just look at it:



The parallel coordinates plot is quite simply revelatory. Where all the other plots tease us with tit-bits of truth, this fellow reveals to us the truth himself. And there is a reason for this: the parallel coordinates plot contains every single bit of data in the iris dataset - nothing is left out.

Okay, so enough with the teacherly enthusiasm, but nonetheless, one must admit that at a mere glance at this plot one can see how similar the versicolors and virginicas are to each other, and just how different they are both from the setosas. They both follow a zigzag pattern, with the virginicas' variables having greater values, particularly in the case of the sepal and petal lengths. Contrast this with the poor setosa, that only has a simpering decline to show for itself.

Finding Correlations Between the Variables

A key component of data analysis is of course the determination of whether or not variables are correlated with each other, and while scatter plots are helpful for visually revealing correlations, in order to quantify correlations one needs to move beyond plots and use math to coerce a value from the data that would somehow represent the extent to which the variables are correlated. One of the mathematical technique for achieving this coercion is called regression analysis, although it might be more helpful to think of it as a progression analysis, i.e. how the variable-values progress with each other: if one variable increases/progresses what happens to the other? Could one go so far as to say that one variable is dependent on another variable? This is what regression analysis is meant to achieve: an understanding of just how much an independent variable affects the value of a dependent variable.

Using Linear Regression

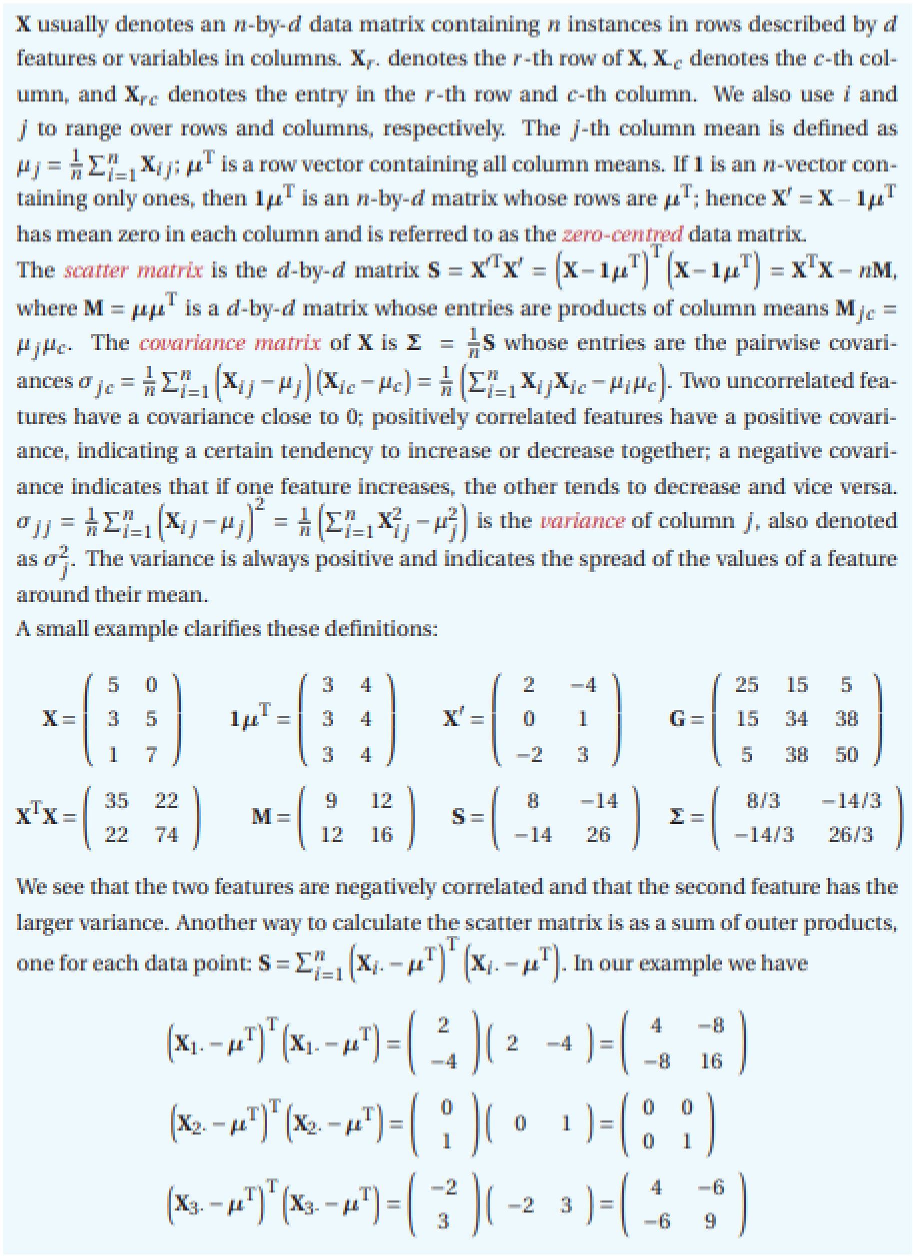
The most common kind of regression analysis is linear regression, which tries to find a straight line (hence 'linear') that describes the relationship between one dependent variable and one or more explanatory variables, although here we will only perform single linear regression, which only uses one explanatory variable. In performing linear regression one will also calculate the variable's covariance, their coefficient of correlation, as well as their coefficient of determination, the latter being the most useful indicator of how closely the variables are related.

It must in a sense remain an open question whether or not in order to perform data analysis to the best of his ability a data analyst needs to understands the mathematical basis of the tools he uses. Programming languages such as Python and R contain packages that allow one to perform complex mathematical manipulations simply by calling ready-made functions. For example, if we want use Python to create a line capable of representing the relationship between two variables, we can simply import the Pandas package and

To better understand what is going on here, I find it helpful to dive into the mathematics. There is a nice, simple explanation of how to create a linear regression plot here. The equation for the regression line is y = b0 + b1 * x, where b0 is the y-intercept and b1 is the slope of the line. b1, known as the regression coefficient, is defined as the covariance of the two variables divided by the variance of the explanatory/independent variable, i.e. its range. b0 is defined as the regression coefficient multiplied by the mean of the independt variable, subtracted from the mean of the dependent variable. In order to calculate the covariance I will manually calculate the covariance matrix for all the variables, which has many uses in data analysis beyond linear regression and is thus useful to understand. Once the covariance matrix is calculated, we can also calculating the coefficients of correlation and determination, which also provide insight into how the variables are related.

We will also want to create the residual plot for each attempt at a linear regression plot. If the y variable value can be 'predicted' with the linear regression equation as b0 + b1 * x, then the residual value is the difference between the actual y value and the predicted y value, and as such the residual plot consists of the explanatory variable values plotted against the magnitude of how much the actual dependent variables values differ from the predicted values (the 'residual' values). We want to make sure that there are no patterns in the residual plot, i.e. that the residuals are not in any way explained by the independent variables, such that their coefficient of determination is as close to zero as possible.

We will perform single linear regression analysis on petal length and petal width without distinguishing between the species, as from the scatter plots above these appear to be the most closely correlated variables. The first step then is to calculate the variables' covariance, the steps of which are as follows (for detailed instructions please see the comments in the 'linearRegression.py' file):



$\mathbf{X}$ usually denotes an n-by-d data matrix containing n instances in rows described by d features or variables in columns. $\mathbf{X}_{r\cdot}$ denotes the r-th row of $\mathbf{X}$, $\mathbf{X}_{\cdot c}$ denotes the c-th column, and $\mathbf{X}_{rc}$ denotes the entry in the r-th row and c-th column. We also use i and j to range over rows and columns, respectively. The j-th column mean is defined as $\mu_j = \frac{1}{n}\sum_{i=1}^{n}\mathbf{X}_{ij}$; $\boldsymbol{\mu}^{\mathrm{T}}$ is a row vector containing all column means. If $\mathbf{1}$ is an n-vector containing only ones, then $\mathbf{1}\boldsymbol{\mu}^{\mathrm{T}}$ is an n-by-d matrix whose rows are $\boldsymbol{\mu}^{\mathrm{T}}$; hence $\mathbf{X}' = \mathbf{X} - \mathbf{1}\boldsymbol{\mu}^{\mathrm{T}}$ has mean zero in each column and is referred to as the *zero-centred* data matrix.
The *scatter matrix* is the d-by-d matrix $\mathbf{S} = \mathbf{X}'^{\mathrm{T}}\mathbf{X}' = \left(\mathbf{X}-\mathbf{1}\boldsymbol{\mu}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X}-\mathbf{1}\boldsymbol{\mu}^{\mathrm{T}}\right) = \mathbf{X}^{\mathrm{T}}\mathbf{X} - n\mathbf{M}$, where $\mathbf{M} = \boldsymbol{\mu}\boldsymbol{\mu}^{\mathrm{T}}$ is a d-by-d matrix whose entries are products of column means $\mathbf{M}_{jc} = \mu_j\mu_c$. The *covariance matrix* of $\mathbf{X}$ is $\boldsymbol{\Sigma} = \frac{1}{n}\mathbf{S}$ whose entries are the pairwise covariances $\sigma_{jc} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{X}_{ij}-\mu_j\right)\left(\mathbf{X}_{ic}-\mu_c\right) = \frac{1}{n}\left(\sum_{i=1}^{n}\mathbf{X}_{ij}\mathbf{X}_{ic} - \mu_i\mu_c\right)$. Two uncorrelated features have a covariance close to 0; positively correlated features have a positive covariance, indicating a certain tendency to increase or decrease together; a negative covariance indicates that if one feature increases, the other tends to decrease and vice versa. $\sigma_{jj} = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{X}_{ij}-\mu_j\right)^2 = \frac{1}{n}\left(\sum_{i=1}^{n}\mathbf{X}_{ij}^2 - \mu_j^2\right)$ is the *variance* of column j, also denoted as σ_j^2. The variance is always positive and indicates the spread of the values of a feature around their mean.
A small example clarifies these definitions:

$$\mathbf{X} = \begin{pmatrix} 5 & 0 \\ 3 & 5 \\ 1 & 7 \end{pmatrix} \quad \mathbf{1}\boldsymbol{\mu}^{\mathrm{T}} = \begin{pmatrix} 3 & 4 \\ 3 & 4 \\ 3 & 4 \end{pmatrix} \quad \mathbf{X}' = \begin{pmatrix} 2 & -4 \\ 0 & 1 \\ -2 & 3 \end{pmatrix} \quad \mathbf{G} = \begin{pmatrix} 25 & 15 & 5 \\ 15 & 34 & 38 \\ 5 & 38 & 50 \end{pmatrix}$$

$$\mathbf{X}^{\mathrm{T}}\mathbf{X} = \begin{pmatrix} 35 & 22 \\ 22 & 74 \end{pmatrix} \quad \mathbf{M} = \begin{pmatrix} 9 & 12 \\ 12 & 16 \end{pmatrix} \quad \mathbf{S} = \begin{pmatrix} 8 & -14 \\ -14 & 26 \end{pmatrix} \quad \boldsymbol{\Sigma} = \begin{pmatrix} 8/3 & -14/3 \\ -14/3 & 26/3 \end{pmatrix}$$

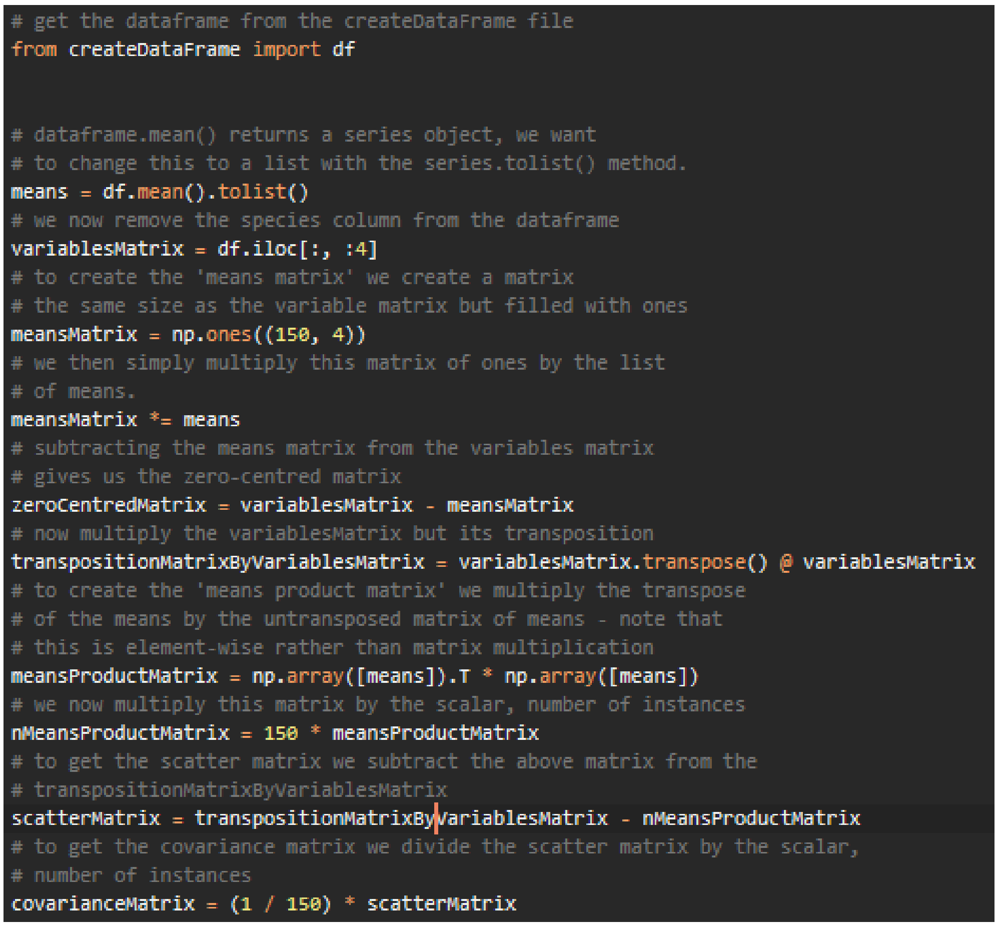
We see that the two features are negatively correlated and that the second feature has the larger variance. Another way to calculate the scatter matrix is as a sum of outer products, one for each data point: $\mathbf{S} = \sum_{i=1}^{n}\left(\mathbf{X}_{i\cdot}-\boldsymbol{\mu}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X}_{i\cdot}-\boldsymbol{\mu}^{\mathrm{T}}\right)$. In our example we have

$$\left(\mathbf{X}_{1\cdot}-\boldsymbol{\mu}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X}_{1\cdot}-\boldsymbol{\mu}^{\mathrm{T}}\right) = \begin{pmatrix} 2 \\ -4 \end{pmatrix}\begin{pmatrix} 2 & -4 \end{pmatrix} = \begin{pmatrix} 4 & -8 \\ -8 & 16 \end{pmatrix}$$

$$\left(\mathbf{X}_{2\cdot}-\boldsymbol{\mu}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X}_{2\cdot}-\boldsymbol{\mu}^{\mathrm{T}}\right) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\left(\mathbf{X}_{3\cdot}-\boldsymbol{\mu}^{\mathrm{T}}\right)^{\mathrm{T}}\left(\mathbf{X}_{3\cdot}-\boldsymbol{\mu}^{\mathrm{T}}\right) = \begin{pmatrix} -2 \\ 3 \end{pmatrix}\begin{pmatrix} -2 & 3 \end{pmatrix} = \begin{pmatrix} 4 & -6 \\ -6 & 9 \end{pmatrix}$$

The script used to calculate the covariance is as follows:



```
# get the dataframe from the createDataFrame file
from createDataFrame import df


# dataframe.mean() returns a series object, we want
# to change this to a list with the series.tolist() method.
means = df.mean().tolist()
# we now remove the species column from the dataframe
variablesMatrix = df.iloc[:, :4]
# to create the 'means matrix' we create a matrix
# the same size as the variable matrix but filled with ones
meansMatrix = np.ones((150, 4))
# we then simply multiply this matrix of ones by the list
# of means.
meansMatrix *= means
# subtracting the means matrix from the variables matrix
# gives us the zero-centred matrix
zeroCentredMatrix = variablesMatrix - meansMatrix
# now multiply the variablesMatrix but its transposition
transpositionMatrixByVariablesMatrix = variablesMatrix.transpose() @ variablesMatrix
# to create the 'means product matrix' we multiply the transpose
# of the means by the untransposed matrix of means - note that
# this is element-wise rather than matrix multiplication
meansProductMatrix = np.array([means]).T * np.array([means])
# we now multiply this matrix by the scalar, number of instances
nMeansProductMatrix = 150 * meansProductMatrix
# to get the scatter matrix we subtract the above matrix from the
# transpositionMatrixByVariablesMatrix
scatterMatrix = transpositionMatrixByVariablesMatrix - nMeansProductMatrix
# to get the covariance matrix we divide the scatter matrix by the scalar,
# number of instances
covarianceMatrix = (1 / 150) * scatterMatrix
```

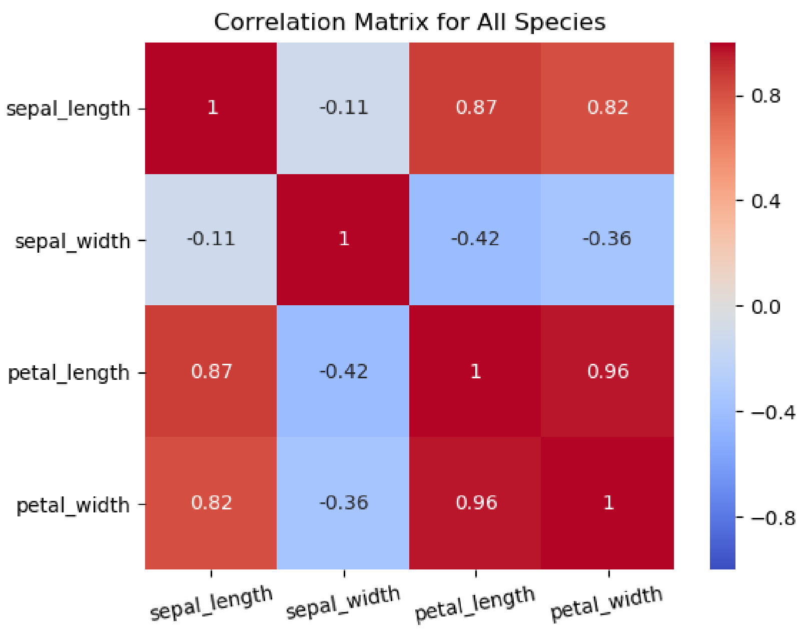
One first calculates the scatter matrix:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	102.168333	-5.8510	189.778667	77.018667
sepal_width	-5.851000	28.0126	-47.935200	-17.579200
petal_length	189.778667	-47.9352	463.863733	193.161733
petal_width	77.018667	-17.5792	193.161733	86.779733

Which is closely followed by the correlation matrix (the diagonals here should of course be one, but this is not the case here due to Python's floating point imprecision):

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	0.993333	-0.108640	0.865942	0.812501
sepal_width	-0.108640	0.993333	-0.417713	-0.354167
petal_length	0.865942	-0.417713	0.993333	0.956339
petal_width	0.812501	-0.354167	0.956339	0.993333

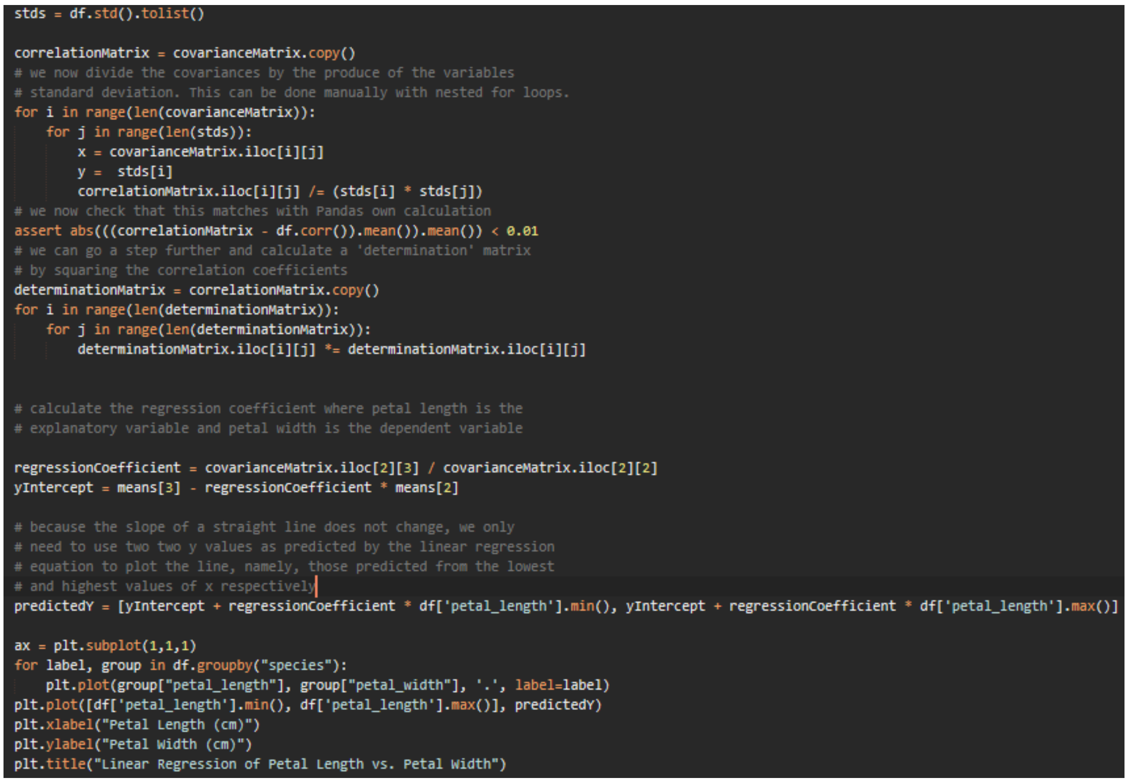
Of course, where performing actual analysis, one would just call the dataframe.corr() method to create the correlation matrix, and then use seaborn's heatmap() method to display it (with a few parameters to tailor it to one's tastee).



And then finally one arrives at the 'determination matrix' (i.e. a matrix containing the coefficients of determination, R2):

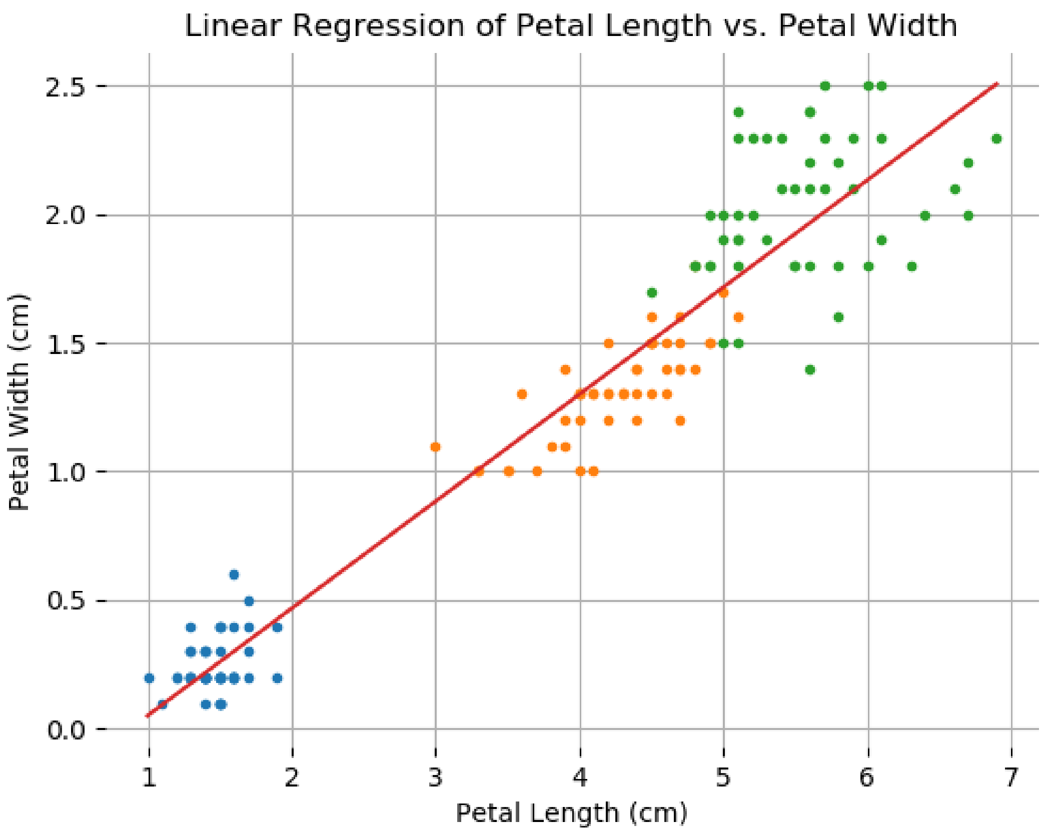
	sepal_length	sepal_width	petal_length	petal_width
sepal_length	0.986711	0.011803	0.749856	0.660157
sepal_width	0.011803	0.986711	0.174484	0.125434
petal_length	0.749856	0.174484	0.986711	0.914584
petal_width	0.660157	0.125434	0.914584	0.986711

Once the covariance matrix is calculated, it is straight forward to calculate the regression coefficient (b1) and the y-intercept (b0):

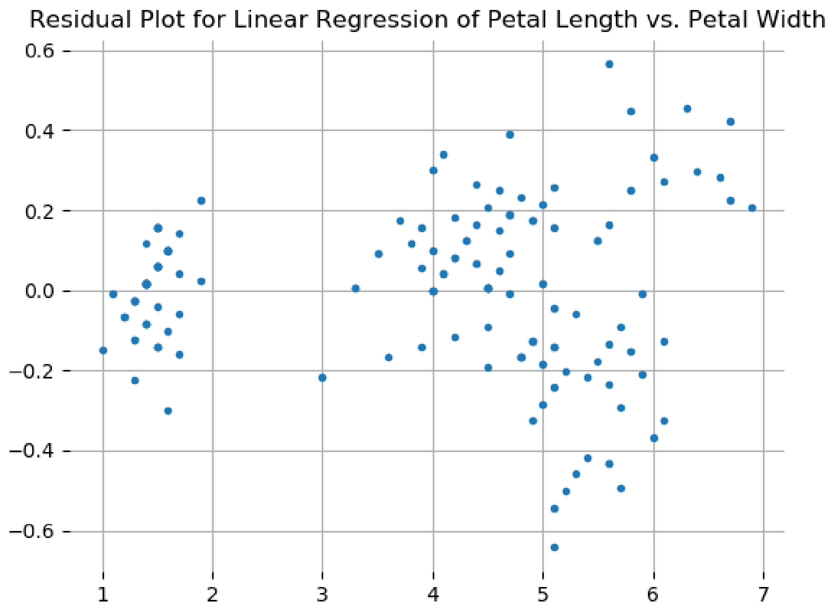


```
stds = df.std().tolist()

correlationMatrix = covarianceMatrix.copy()
# we now divide the covariances by the produce of the variables
# standard deviation. This can be done manually with nested for loops.
for i in range(len(covarianceMatrix)):
    for j in range(len(stds)):
        x = covarianceMatrix.iloc[i][j]
        y =  stds[i]
        correlationMatrix.iloc[i][j] /= (stds[i] * stds[j])
# we now check that this matches with Pandas own calculation
assert abs(((correlationMatrix - df.corr()).mean()).mean()) < 0.01
# we can go a step further and calculate a 'determination' matrix
# by squaring the correlation coefficients
determinationMatrix = correlationMatrix.copy()
for i in range(len(determinationMatrix)):
    for j in range(len(determinationMatrix)):
        determinationMatrix.iloc[i][j] *= determinationMatrix.iloc[i][j]


# calculate the regression coefficient where petal length is the
# explanatory variable and petal width is the dependent variable

regressionCoefficient = covarianceMatrix.iloc[2][3] / covarianceMatrix.iloc[2][2]
yIntercept = means[3] - regressionCoefficient * means[2]

# because the slope of a straight line does not change, we only
# need to use two two y values as predicted by the linear regression
# equation to plot the line, namely, those predicted from the lowest
# and highest values of x respectively
predictedY = [yIntercept + regressionCoefficient * df['petal_length'].min(), yIntercept + regressionCoefficient * df['petal_length'].max()]

ax = plt.subplot(1,1,1)
for label, group in df.groupby("species"):
    plt.plot(group["petal_length"], group["petal_width"], '.', label=label)
plt.plot([df['petal_length'].min(), df['petal_length'].max()], predictedY)
plt.xlabel("Petal Length (cm)")
plt.ylabel("Petal Width (cm)")
plt.title("Linear Regression of Petal Length vs. Petal Width")
```

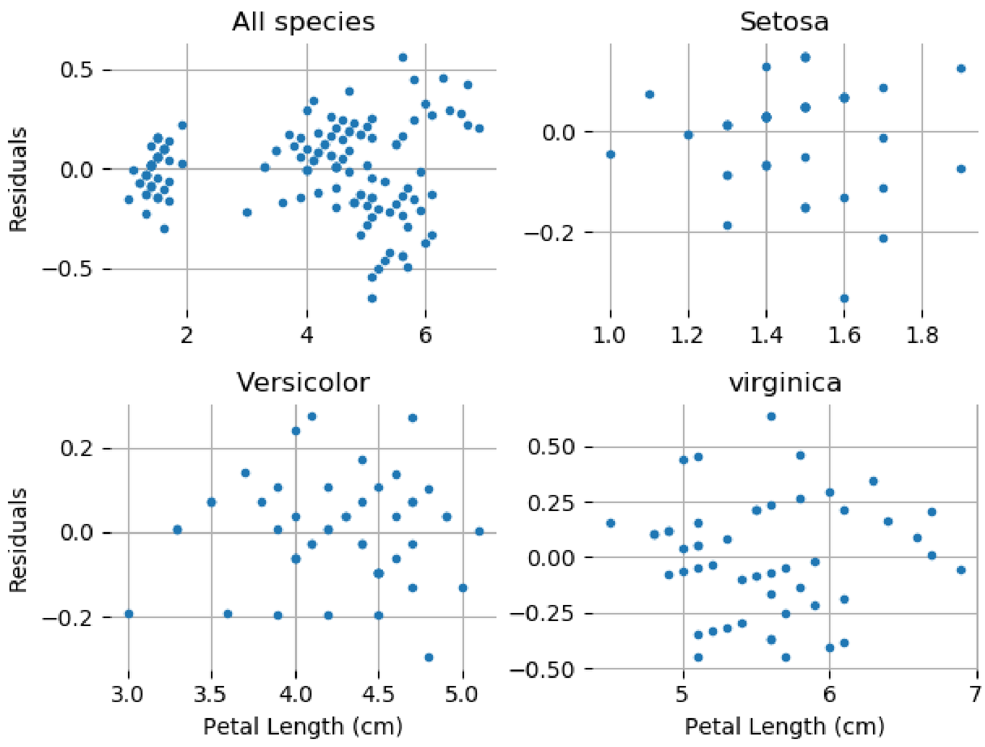
And finally the following linear regression plot is achieved:



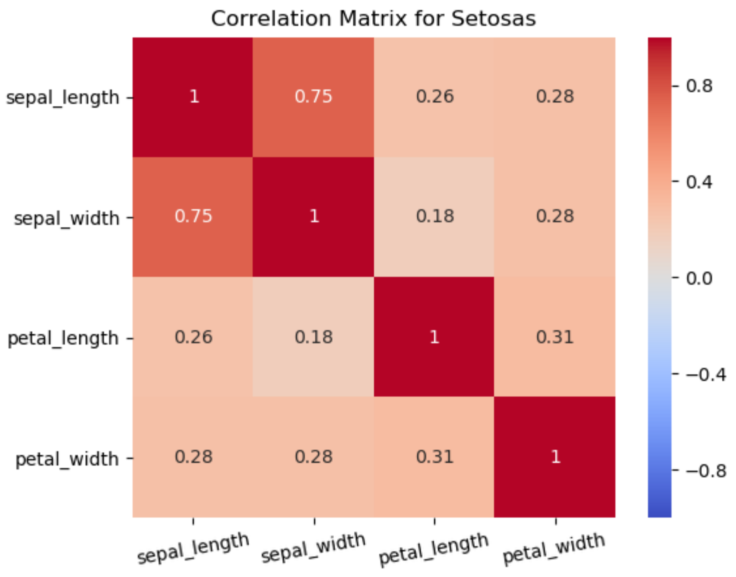
Along with the following residual plot:



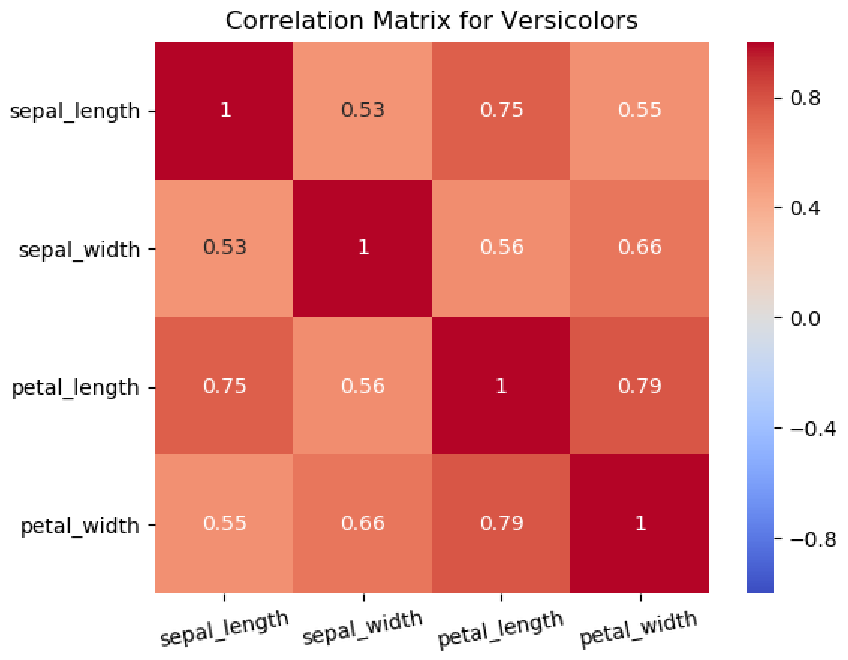
The above residual plot is perhaps the most fascinating of all those so far shown, precisely because of its ambiguity. We would like it to reveal no pattern (or more particularly for every 'section' of the x-axis, the y-values should be evenly distributed above and below zero), and yet for petal lengths of between five and six cm most of the residuals are negative, while for the rest of the petal lengths there appears to be a slight upward trend such that the larger the petal length the greater the residual. This does not mean that petal length and petal length are not correlated, but rather it suggests that there is some other factor that is determining petal width rather than petal length (it could also suggest that a linear fit is not appropriate, and that a nonlinear fit would be better, but that is unlikely the case here). It is interesting to note that if one looks at the scatter plot for petal length vs petal width, above, the only petal lengths of between five and six cm are virginicas, which lends credit to the idea that there is some factor other than its petal length that determines an iris flower of 5-6 cm petal length's petal width, namely, the species, which tells us that linear regressions for each species would probably be more successful. It would be helpful then to compare the residual plot for the iris flowers altogether with the individual plots for the species:

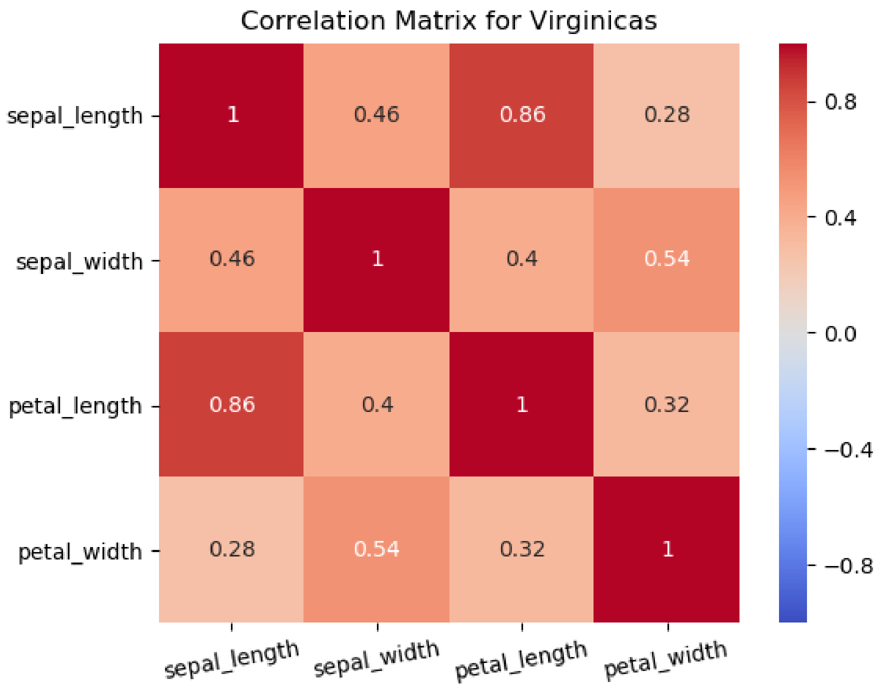


There does not appear to be any significant patterns in any of the individual plots, which again tells us that in analyzing the iris dataset for correlations it would be more profitable to analyze the classes separately. And yet if one considers the correlation matrices for the individual species, something strange is revealed:


Correlation Matrix for Setosas

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1	0.75	0.26	0.28
sepal_width	0.75	1	0.18	0.28
petal_length	0.26	0.18	1	0.31
petal_width	0.28	0.28	0.31	1





From these we can see that while across all species petal length and petal width have a correlation coefficient of 0.96, for Setosa, Versicolors and Virginicas, the coefficients are 0.31, 0.79 and 0.32 respectively. Somehow the correlation between petal length and width is greater when the species are not considered. Indeed this is true for all of the variables except sepal width, which is only variable to have a negative correlation with the other variables in the overall correlation matrix. What then is so special about sepal width?

The reason for sepal width's negative correlations in the overall matrix is that while Setosa have the highest sepal width values, they have the lowest values for all the other variables, such that in the overall scatter plots for sepal width vs the other variables, as sepal width increases the other variables actually tend to decrease, due to the setosas' influence. And yet there is another interesting feature of sepal widths: if you look at the scatter plots, sepal width actually quite clearly stands out as the variable that is least linked with species. Sepal width presents us with a kind of paradox then: it is the variable least linked with species variables, and yet it is the only variable whose correlations do not improve if one unlinks them from the species variables. Of course, this is only an ostensible paradox, and is accounted for by the fact that the correlations depend on the other variables also, and as said before the setosa's slightly higher-valued sepal width and lower valued everything else then brings about negative correlations.

It also interesting perhaps to note that the Setosas' variables are generally very weakly correlated, although this could be explained by the fact that their values are quite small and also have a small variance, so perhaps if the measurements were more accurate, i.e. if they were measured to two or three places after the decimal point, the correlation would improve.

By far the more interesting thing to note, however, is the fact that apart from sepal width the variables' correlations improve as you abstract from the species, and again while this might seem like a paradox, in effect this serves as a warning not to mix correlation with classification, i.e. just because something is a class does not mean that its variables will be highly correlated. In the case of Iris flowers, it is clear that the classification was not in terms of correlations between variables, but rather in terms of variable ranges. And that leads us nicely into our next section. We can think of it this way: iris flowers are a class of flowers, and stenosis, Versicolor and Virginias are classes of iris flowers; while iris flowers show strong correlations amongst sepal length, petal length, and width, but with gaps in the ranges of values for petal length and width, iris species show weaker correlations amongst the same variables, but are better able to account for the variables' ranges of values. We could conclude from this that the Iris designation accounts for strong variable correlation, whereas the species designations account for the actual variable values.

And with this, we can nicely move on to classification.

Statistical Classification of the Data

As mentioned above, one of the reasons - if not the reason - for the Iris dataset's continued popularity as an educational tool, is the fact that it was the dataset that Fisher worked with to develop Linear Discriminate Analysis (LDA), which is a statistical classification technique.

Linear Discriminant Analysis of the Dataset

Linear discriminant analysis (LDA), normal discriminant analysis (NDA), or discriminant function analysis is a a method used in statistics, pattern recognition, and machine learning to find a linear combination of features that characterizes, i.e. is capable of discriminating between, two or more classes of objects or events [3] (in the case of the Iris dataset we are looking to discriminate between the species). In mathematics, a linear combination is an expression constructed from a set of terms by multiplying each term by a constant and adding the results (e.g. a linear combination of x and y would be any expression of the form ax + by, where a and b are constants) [4]. How exactly linear combinations are come upon by LDA will be addressed below, but for now we can simply think of the results of LDA as discriminants.

It is important to note that while LDA is often refered to as a kind of dimensionality reduction, it is in reality much more than that. Dimensionality reduction simply consists in taking two or more dimension (i.e. properties of an object that once plotted each become an axis), and reducing them to a lesser amount of dimensions, typically one or two, while retaining as much of the information contained in the original dimensions as possible. The reason for doing this is that multi-dimensional data, for example four-dimensional data as in the case of the Iris dataset, is very different for humans to get their heads around. While a human mind reasonably well trained in pattern-finding can look at a 2D plot and see roughly what is going on, anything more than 2D presents a challenge. The solution to this is thus to reduce the three dimensions to two, such that minimal information is lost in this reduction and the human eye can more readily see patterns in the data.

It is also important to note, however, that dimensionality reduction in itself is meaningless. One could easily, for example, take the variables of each flower in the Iris Dataset and sum them together, and one would have thus performed a dimensionality reduction. Of course, a dimension that consists in the sum of the original dimensions is unlikely to be meaningful. The question is thus, in which cases can dimensionality reduction be meaningful? There are generally three:

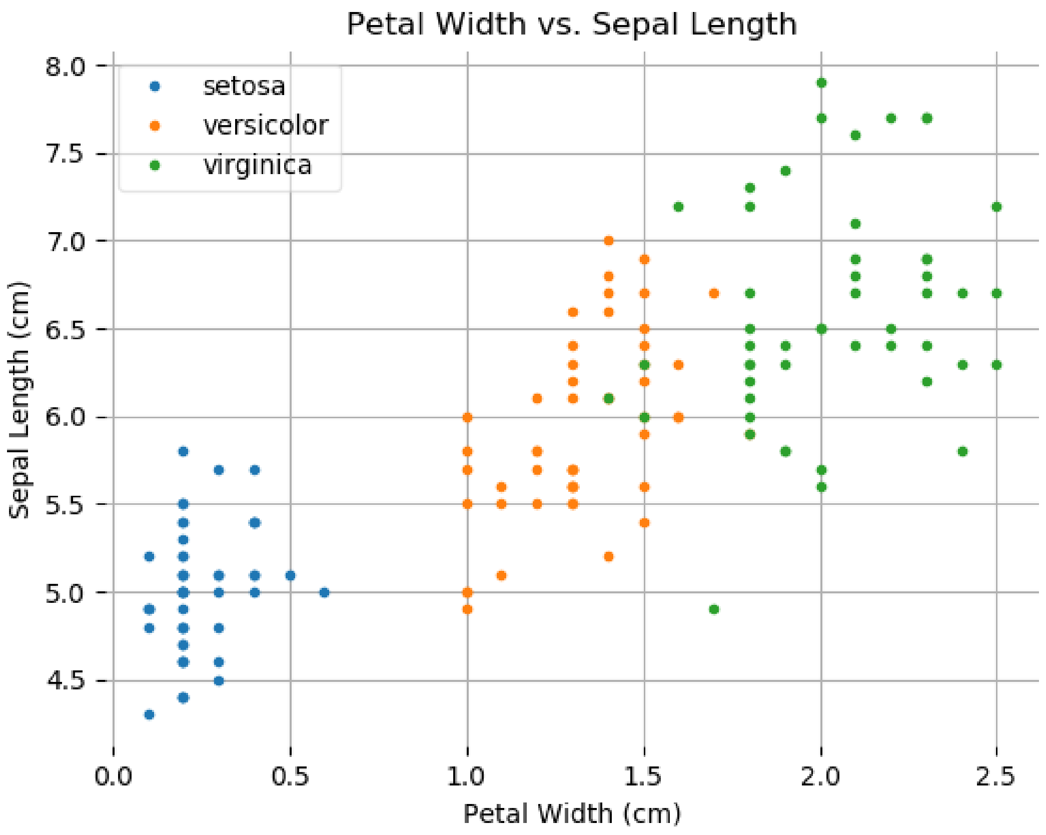
1. The first case is not of meaning, but of use. Reducing an object's dimensionality can make render it much less resource-intensive to include in computations. If one has a massive dataset of millions of objects with hundreds of dimensions each, reducing these dimensions would make it much more feasible to perform computations on the data.
2. The second case is more clearly about meaning. As already said above, the human eye cannot grasp multi-dimensional data easily. By reducing the data's dimensionality to a more human-manageable number (generally 2D), there is a much greater change of one being able to see patterns in the data. The challenge here then is reducing the dimensions into a dimension that is still meaningful, and that leads us onto the third case.
3. The third case is generally associated with machine learning, and in many ways it is a subcase of the second case. Essentially we want here to reduce the data to a dimension (or a small number of dimensions) that will make it possible to discriminate between the data or the classes in the dataset based on that (or those) dimensions. This dimension is what I (but not the literature at large) refer to in the case of class-discrimination as the discriminant. Not that I distinguish here between dimensionality reductions intended to allow discrimination between the data and those intended to allow discrimination between the classes; there are different sets of algorithms for both of these tasks, the simplest being Principal Component Analysis (PCA) and Linear Discriminate Analysis (LDA) respecively.

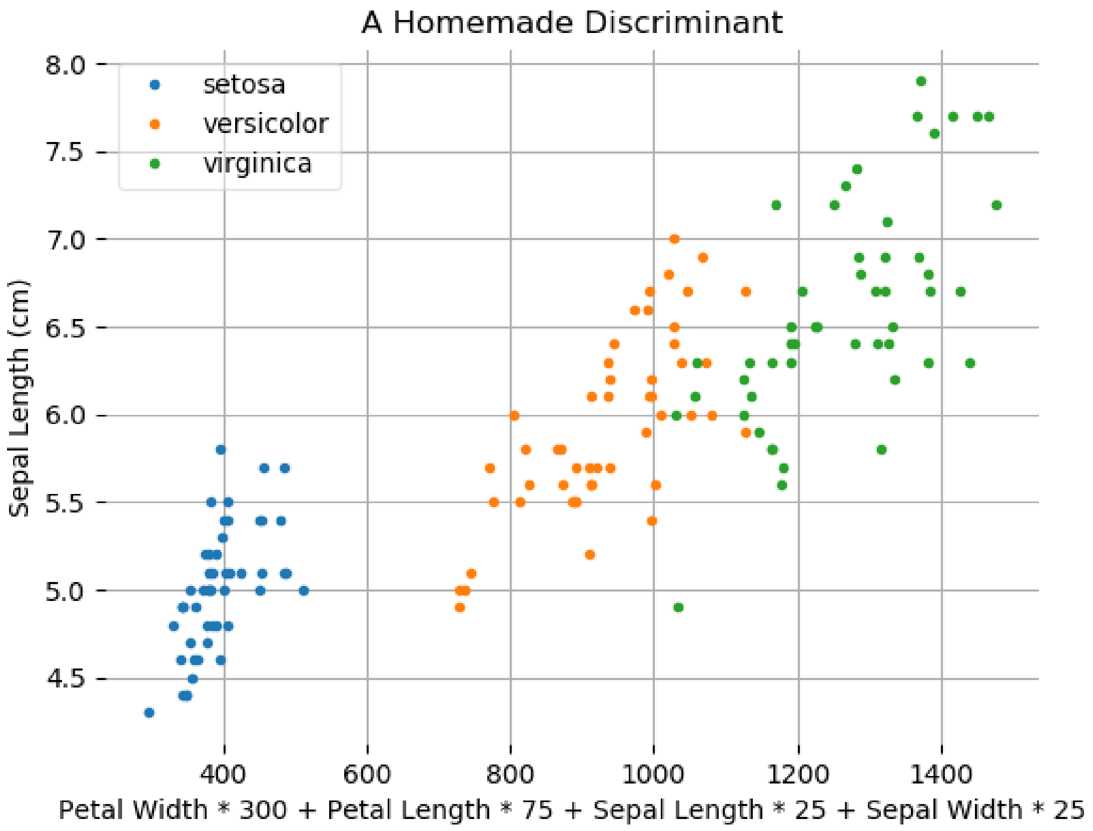
What you will notice about the three cases is that they extend each other in the order I have presented them, such that the third case is always at the same time an instance of the second and first cases, while the second case is not necessarily also an instance of the third case, and the first cases is not necessarily an instance of the second or third cases, i.e. one can perform a dimensionality reduction for computational-efficiency reasons (case one) without finding a discriminant (case three), but one cannot find a discriminant without performing a dimensionality reduction that also happens to allow for more efficient computation of the data.

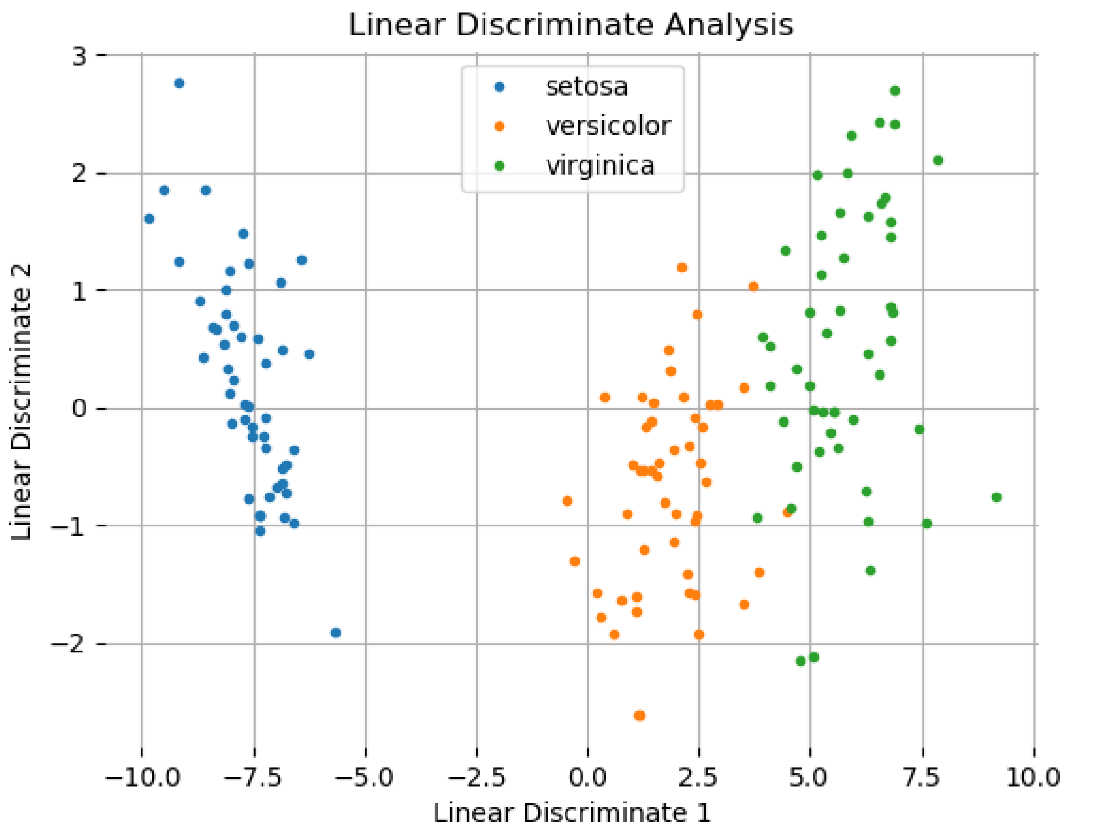
LDA then, is an example of the third case where a discriminant is sought, so by necessity it is also a example of the first case. Note the LDA seeks a linear discriminant. The meaning of 'linear' here comes from linear combination.

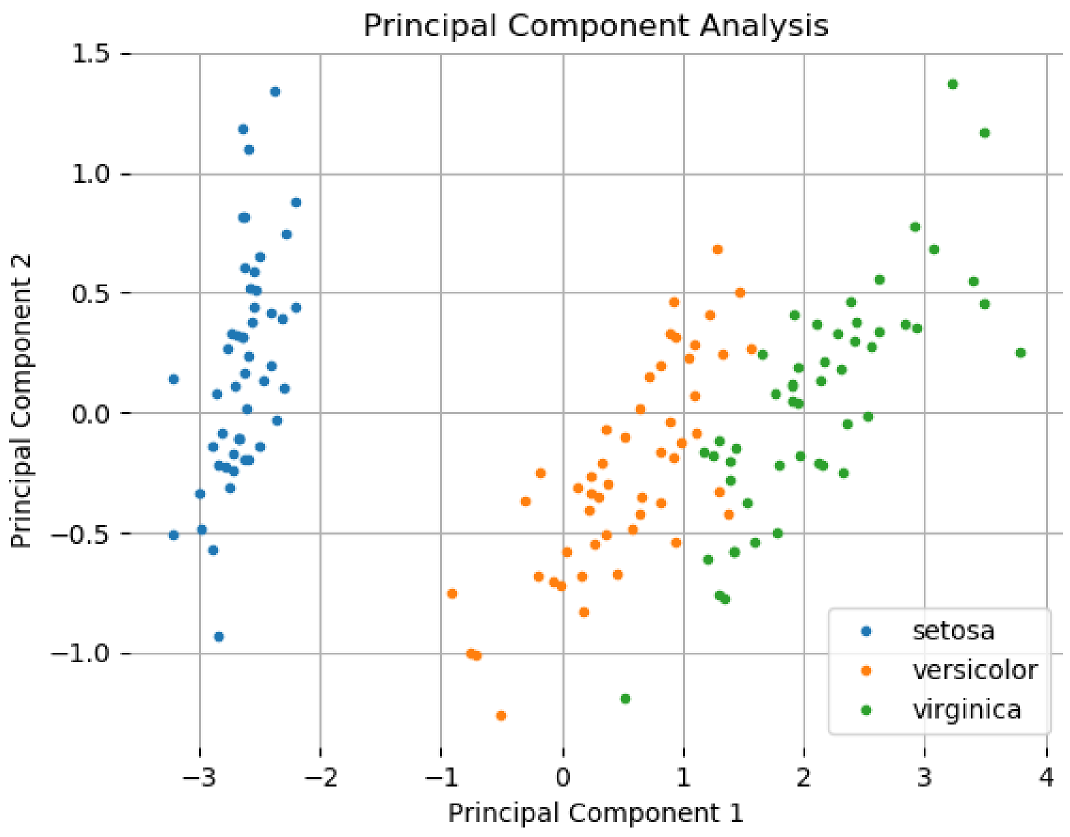
Before we dig into the mathematics of how exactly LDA works, t is helpful when trying to understand what exactly LDA achieves in terms of machine learning to compare its results (i.e. the linear discriminants) with three other possible discriminants:

1. The variable in the dataset best able to discriminate between the species, which from the plots above would appear to be petal width.
2. A rudimentary 'homemade' attempt at reducing the flower's dimensions by weighting the dimensions in terms of their ability to allow us discriminate between the classes, i.e. petal width having the most weight, then petal length, and then the sepal values:
3. The results of a Principal Component Analysis (PCA) of the dataset. PCA is similar to LDA except that instead of trying to create a dimension that maximizes the distance between the classes, it tries to create a dimension that maximizes the total distance between the data points themselves. Thus one might say that where LDA creates a class-discriminatn, PCA creates a data-discriminant. The mathematical difference between LDA and PCA will be furthered explained below.




A Homemade Discriminant
setosa
versicolor
virginica
Sepal Length (cm)
Petal Width * 300 + Petal Length * 75 + Sepal Length * 25 + Sepal Width * 25
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
400
600
800
1000
1200
1400


Linear Discriminate Analysis
setosa
versicolor
virginica
Linear Discriminate 2
Linear Discriminate 1
3
2
1
0
−1
−2
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0



We can see quite clearly that the first linear discriminant (LD1) is better at distinguishing between the species than both petal width, our homemade attempt, and even PCA. Interestingly, the PCA appears to offer the worst results, and our homemade attempt doesn't appear to be any better than petal width. In terms of petal width and the actual linear discriminant, both can successfully distinguish between the setosas, but whereas in the case of petal width, there are several versicolor values that are greater than several virginicas, in the case of LD1 the only overlap between the species is one versicolor value that is greater than six virginicas (there is perhaps one other versicolor value that is greater than one other viriginica).

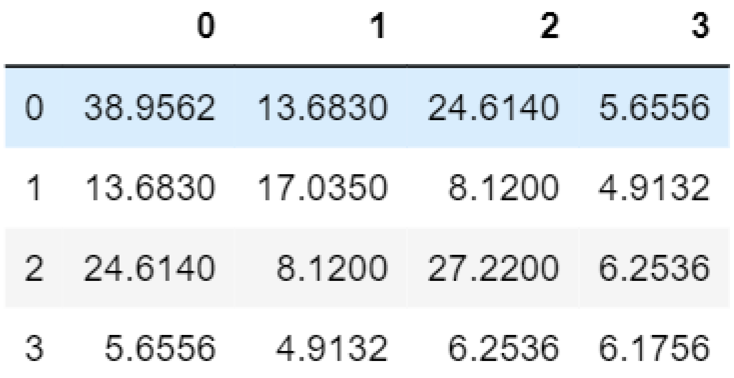
Clearly then, LDA is a good candidate for discriminating between the species, and thus also for teaching a machine how to determine what species an iris flower is based on its petal and sepal length and width. The better performance of LDA here in comparison to PCA also demonstrates the difference between supervised and unsupervised learning techniques. In a dataset such as this where the classes are already known, LDA, a supervised learning technique, generally outperforms unsupervised techniques such as PCA.

How then does one perform linear discriminant analysis? Well, with Python packages, one just has to import and call the appropriate functions, but mathematically the steps are as follows:

1. Calculate the within-class scatter matrix (in PCA we would only calculate the scatter matrix regardless of class, but after the next step the process is the exact same).
2. Calculate the between-class scatter matrix.

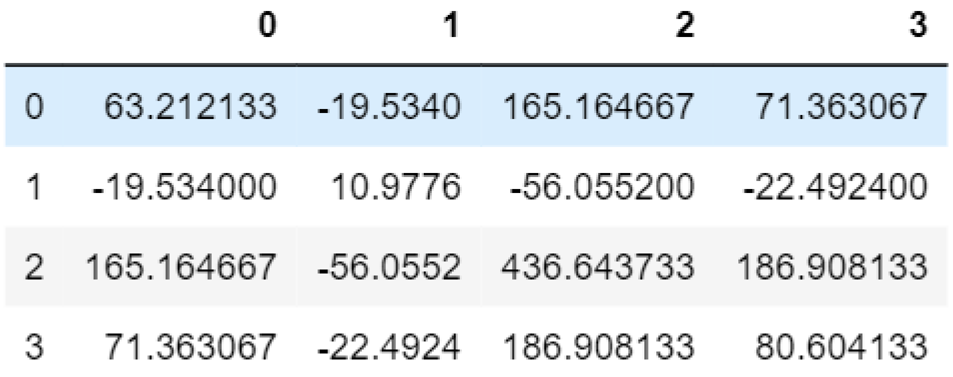
3. Decompose the product of the inverse of the within-class matrix and the between-class matrix into its pairs of eigenvalues and eigenvectors. The most intuitive meaning of eigenvalues is that the total value of all the dataset's eigenvalues is capable of representing the variance of the datasets values, such that each eigenvalue is only capable of representing a portion of that variance. If we divide an eigenvalue by the total value of the eigenvalues the result is an eigenvalues 'explained variance,' which is the proportion (on a scale of 0-1) of the dataset's variables' variance that is explained by the eigenvector. This is referred to as η 2, and is the equivalent of what in linear regression is known as the coefficient of determination . We basically want to find particularly high-value eigenvalues whose corresponding eigenvectors will be capable of transforming the dataset to a lower dimension while still being able to representing a sufficiently large amount of the dataset's variables' variance. The eigenvectors are in effect the linear discriminants we are looking for.
4. Once we have identified the highest eigenvalues we are interested in, we create a matrix with their corresponding eigenvectors and then multiply the dataset-matrix by this to achieve the dimensionality reduction to the linear discriminant we have chosen. More detailed explanation of how to complete each of these steps can be found in the comments of the python file, 'dimensionalityReduction.py'.

One first calculates the within-class scatter matrix:



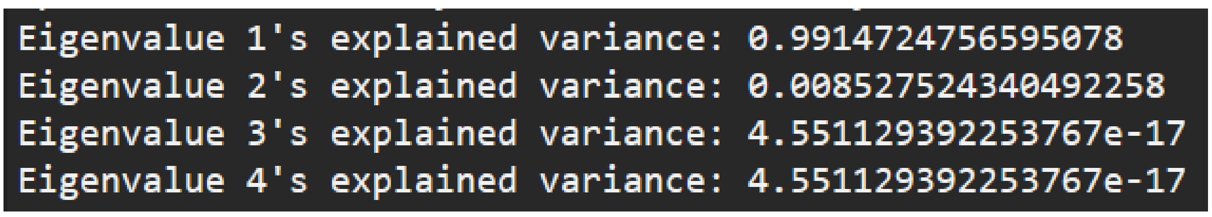
	0	1	2	3
0	38.9562	13.6830	24.6140	5.6556
1	13.6830	17.0350	8.1200	4.9132
2	24.6140	8.1200	27.2200	6.2536
3	5.6556	4.9132	6.2536	6.1756

Followed by the between-class scatter matrix



	0	1	2	3
0	63.212133	-19.5340	165.164667	71.363067
1	-19.534000	10.9776	-56.055200	-22.492400
2	165.164667	-56.0552	436.643733	186.908133
3	71.363067	-22.4924	186.908133	80.604133

Once the eigenvalues have been calculated, their 'explained variance' values should be calculated to determine which eigenvalues should be used.

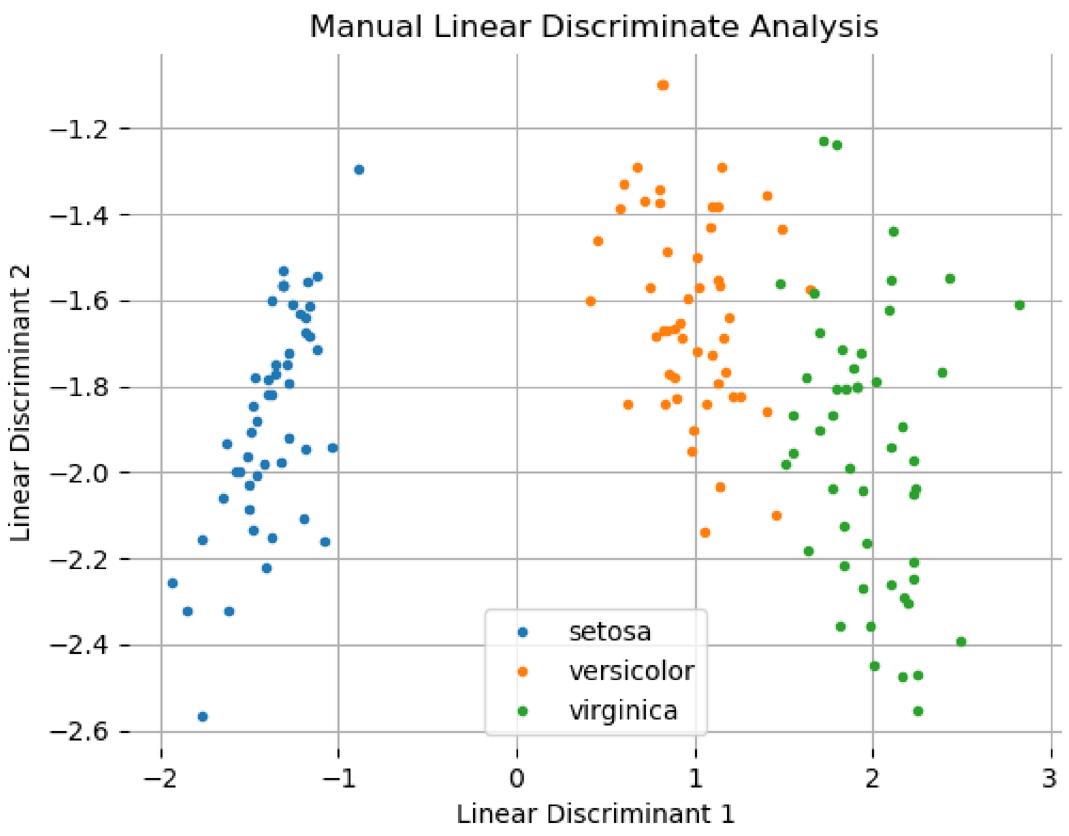


```
Eigenvalue 1's explained variance: 0.9914724756595078
Eigenvalue 2's explained variance: 0.008527524340492258
Eigenvalue 3's explained variance: 4.551129392253767e-17
Eigenvalue 4's explained variance: 4.551129392253767e-17
```

Once one has chosen the eigenvalues, one then creates a matrix from their eigenvectors that will transform the dataset:

å

And here is the plot I arrived at:



Note that the scale and even the shape of this LDA plot is different from the one above that was created with Python's sklearn package. This is not surprising considering Python's floating-point imprecision and the amount of steps involved in the process. Note however that the manually created plot is just as successful as distinguishing between the versicolors and virginicas, which is exactly what LDA is supposed to achieve.

Of course, when evaluating machine learning models, one does not simply glance over such plots; one must quantify their success. Sklearn provides two standard ways of doing just that: confusion matrices and classification reports. Confusion matrices are matrices where the rows represent the actual classes of the of the dataset's instances and the columns represents the classes that the model predicted, such that the number of true positive predictions is represented along the diagonal. The following confusion matrix is of sklearn's LDA model trained on 80% of the Iris dataset and tested on the remaining 20%. The first row represents the setosas, the second the versicolor, and the third the virgininicas that were tested. The first column represents the instances that the model predicted to be setosas, the second column versicolors and the third column virginicas. The only incorrect prediction was a virginica predicted to be a versicolor:

```
[[10  0  0]
 [ 0  8  0]
 [ 0  1 11]]
```

The classification report displays the model's precision, recall, f1-score and support for each class, as the macro and weighted average of each of the aforementioned scores.

- Precision is calculated as the percentage of predicted positives that were actually true positives (TP / TP + FP)
- Recall is calculated as the percentage of positive correctly identified as positives (TP / TP + FN).
- F1-score is a weighted average of precision and recall (2 * (precision*recall / precision + recall)).
- Support refers to the number of instances of the particular class.
- The 'macro' average refers to the average without regard to how many instances there are of each class.
- The weighted average refers to the average that takes into account the numbers of each class, such that classes with high number of instances will have a greater weight when the average is calculated.

The following classification is again of sklearn's LDA model trained on 80% of the Iris dataset and tested on the remaining 20%:

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       0.89      1.00      0.94         8
   virginica       1.00      0.92      0.96        12

    accuracy                           0.97        30
   macro avg       0.96      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
```

Conclusion

The Iris dataset remains relevant due to its beginner-friendly nature: it has few features, classes and samples, though those few features and classes interrelate in interesting ways. It is a particularly useful dataset in terms of introducing students to the basic machine learnings concept of differenting classes based on a linear combination of features. What is surely most interesting about the dataset, however, is how it reveals the different characteristics of the classification of a flower as an iris flower and the classification of an iris flowers as either a setosa, a versicolors or a virginica. As stated above, the Iris classification accounts for strong variable correlation, whereas the species classification is generally capable of determing the actual variable values. This should tell us that just as correlation does not imply causality, neither does classification imply correlation.

Appendix

Definition of Key Terms

Coefficient of determination
To understand the reason for the existence of the coefficient of determination () and how it is calculated it is best to first understand covariance and correlation. The coefficient of determination is calculated as the result of squaring the coefficient of correlation, r, and as such is is known as R2 (the difference between r and R is another matter). It has a value between zero and one, and that value multiplied by one hundred represents the percentage of the dependent variable's value that is determined by the explanatory variable. The coefficient of determination is particularly useful when performing multiple linear regression analysis, where the coefficient of correlation loses much of its intuitive meaning. For example, when trying to understand how height affects weight the correlation coefficient does quite well. However, if we want to understand how both height and age taken together affect weight, we are really finding out how a linear combination of height and age affects weight, and it wouldn't make much intuitive sense to find the correlation coefficient in order to be able to say that as the linear combination increases so too does weight; it would make much more sense to find the coefficient of determination in order to be able to say that the linear combination oof height and age is capable of explaining whatever percentage of weight.

Correlation
Correlation is best understood as the result of a mathematical operation applied to a variable's covariance, namely dividing the covariance by the product of each of the variables' standard deviation. The result is called the correlation coefficient, r, which will always have a value of between negative and positive one, with zero meaning no correlation, one meaning perfect positive correlation, and negative one meaning perfect negative correlation. Importantly, if we square the correlation coefficient we get the coefficient of determination

Covariance
Covariance describes how two variables change with each other, more particulalry how a change in one variable (the independent variable) affects the other variable (the dependent variable). Note,

however, that the magnitude of the covariance does not really mean anything, it is only the sign that matters. cov(x, y) = 2 does not at all mean that if x increases by one y should increase by two, it only means that as x increase y should also increase. A negative covariance means that as x increases y increases. In order to understand how tightly variables are related to each other one must calculate their correlation, whose magnitude, unlike that of covariance, does have significance.

Dimensionality Reduction
Dimensionality reduction simply consists in taking two or more dimension (i.e. properties of an object that once plotted each become an axis), and reducing them to a lesser amount of dimensions, typically one or two, while retaining as much of the information contained in the original dimensions as possible.

Discriminant
Discriminant is not a standard term in data analysis, probably because it is a standard term in mathematics and thus there would be room for confusion. But it is useful to understand the 'discriminate' part of linear discriminate analysis. Basically, a discriminant can be thought of as a property (variable) of an object that allows us to determine what class that object belongs to, i.e. we can discriminate between classes based solely on the discriminatn variable (in the case of the Iris Dataset, the three species are the classes). A perfect discriminant would be such that by looking at the value of the discriminant for each object in a dataset we would be able to separate the objects accurately into their different classes. Of course, the 'ready-made' variables of data-sets (such as the iris dataset's petal lengths) are rarely reliable discriminants. In practice, variables have to be combined to produce an accurate discriminant. For more on how this is achieved, please see linear combination, and for additional context, linear discriminate analysis.

Linear Combination
(Please first read the definition of discriminant.) In mathematics, a linear combination is an expression constructed from a set of terms by multiplying each term by a constant and adding the results (e.g. a linear combination of x and y would be any expression of the form ax + by, where a and b are constants) [4].

Linear Discriminate Analysis
(Please first read the definitions of discriminant and linear combination.) Linear discriminant analysis (LDA), normal discriminant analysis (NDA), or discriminant function analysis is a method used in statistics, pattern recognition, and machine learning to find a linear combination of features that characterizes, i.e. is capable of discriminating between, two or more classes of objects or events [3].

Linear Regression
In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression. [9]

Multivariate
Multivariant refers to the property of having multiple variables. Hence a multivariate dataset is a dataset with multiple variables, and multivariate statistics is the statistics of multivariate datasets.

Normal Distribution
A normal distribution is an arrangement of a data set in which most values cluster in the middle of the range and the rest taper off symmetrically toward either extreme [8]. Note, however, that while all normal distributions are bell curves, not all bell curves are normal distributions. Normal distributions are "normal" because according to the central limit theorem, under some conditions the average of many samples (observations) of a random variable with finite mean and variance is itself a random variable whose distribution converges to a normal distribution as the number of samples increases.

Principal Component Analysis
Principal Component Analysis (PCA) is defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some scalar projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on [10]. PCA performs a dimensionality reduction such that each new dimensions is the next best possible basis upon which the datapoints can be separated.

Statistical Classification
In machine learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known [7].

Supervised learning
Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data. This in contrast to unsupervised learning that does not have the luxury of data that is labelled [5].

Unsupervised learning
Unsupervised learning is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision. This in contrast to supervised learning that usually makes use of human-labeled data [6].

References

[1] https://en.wikipedia.org/wiki/Iris_flower_data_set
[2] https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
[3] https://en.wikipedia.org/wiki/Linear_discriminant_analysis
[4] https://en.wikipedia.org/wiki/Linear_combination
[5] https://en.wikipedia.org/wiki/Supervised_learning
[6] https://en.wikipedia.org/wiki/Unsupervised_learning
[7] https://en.wikipedia.org/wiki/Statistical_classification
[8] https://www.tutorialspoint.com/statistics/normal_distribution.htm
[9] https://en.wikipedia.org/wiki/Linear_regression
[10] https://en.wikipedia.org/wiki/Principal_component_analysis
[11] https://machinelearningmastery.com/implement-simple-linear-regression-scratch-python/
[12] Peter Flach. Machine Learning, The Art and Science of Algorithms that Make Sense of Data, 2012

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel,
M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Ma-
chine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.