Repeat Programming for Data Analytics

Mary McHale - G00253431

Background of pandas:

This project requires writing a pandas tutorial in jupyter notebook. It will cover the following:

- How to install pandas in Python.
- Functionality of pandas (Implemented using Iris Dataset).
- How to use pandas.

- Pandas is like Excel in Python [1].
- It is an open source tool for data analysis which conceals all complex calculations.
- It is a Python package which provides quick adaptable and expressive data structures specifically planned to enable working with “relational” or “labeled” data without much effort and easy to understand. Relational data describes data aranged into tables. Labeled data describes the end result after providing headings to data so that it is organised.

Panda intends to be the under-lying foundation for doing concrete, business-like and practical data analysis in Python.

How to install pandas in Python.

Anaconda is used to complete the installation. After installing Anaconda, you may open the conda prompt and run the following command. [2]

```
conda install -c anaconda pandas
```

Part way through the installation, there is a screen prompt - `Proceed (y\n)?`

Type y for the installation to complete.[1]

To ensure that the latest version of pandas is being used, the following code needs to be typed into the command line. `pip install pandas` (if pandas is not downloaded)

`python -m pip install --upgrade pandas` (to upgrade a downloaded version)

Then within jupyter notebook, import the pandas package as `import pandas as pd` (pd is an alias) and to check the current version: `pd.__version__`

Functionalities of Pandas

Pandas is compatible with lots of different types of data. There are two primary data structures of pandas and these accommodate most of the requirements in finance, statistics, social science, and lots of sections of engineering.

- Series (1-dimensional)
- DataFrame (2-dimensional).

The primary source for scientific computing in Python is NumPy (Numerical Python)[3]. As pandas is constructed on top of NumPy, it therefore works well within a scientific computing environment with many other 3rd party libraries. Pandas performs best with small data (in the range 100MB up to 1GB)[4].

*Tabular data with heterogeneously-typed columns as in an SQL table or Excel Ordered and unordered (not necessarily fixed-frequency) time series data.

Primary Funtions: [5] [6]

- Reading data. Commands or code is: read_csv, read_excel

  Example of code: `data = pd.read_csv('my_file.csv')`

In addition, it can copy data from Excel or the web. Comands or code: read_clipboard, read_sql

- Writing data. Strong import and export tools to load data from flat files (CSV and delimited), databases, files from Excel. Commands or code is:

  `.to_excel, .to_json, .to_pickle since .to_csv`

  Example of code:

  `data.to_csv('my_new_file.csv', index=None)`

In aditions, it allows pasting of results from Python to Excel Comands or code: .to_clipboard

- Checking data shape and description Commands or code is:

`data.shape` : This will show the number of rows and columns

`data.describe()` : This will show mean,std,min,max,count and the quartile ranges

- Seeing the data. Commands or code is:

`data.head(10)` : This lists the first 10 (default) rows of data. any number cab be specified here.

The similar command is:

`data.last(10)` : lists the last 10 rows of data.

These are useful data checks to ensure that the working file is as expected and that no data is missing.

- To print a specific row of the data the command is:

`data.loc(3)` which would print the third row or another function called iloc can be used like:

`data.iloc[0:3,0:3]` : will print first 3 rows and 3 column.

Some other possibilities with this are:

`data.iloc[0:5,2:3]` : print rows 0-5 and columns 2-3.

`data.iloc[4:,6:]` : Print all rows starting from row 4 and print all column starting from column 6.

- Use of Logical operations The data can be sub divided into different parts based on using logical operations.Therefore use of "&" (AND) in code, "~" (NOT) and "|" (OR), by adding “(“ and “)” before and after the logical operation.

Example of code:

```
data[(data['column_1']=='french') & (data['year_born']==1990)]
```

This command would return details of individuals was both French and born in 1990.

- Basic plotting

Basic plotting is achieved in pandas using the matplotlib package which is availabel directly in pandas.

Command or code is: `data.plot()`

Example of code:

```
` data['column_numerical'].plot()

  data['column_numerical'].hist() `
```

This displays the output as a histogram.

- Updating the data

Pandas allow for columns to be deleted and inserted from DataFrame. Example of code:

```
data.loc[10, 'column_1'] = 'irish'
```

This will replace the content of the tenth (10th) row in the column_1 with the word 'irish'.

- Converting a row or a column from the dataframe into a list: Command: `data['row1].tolist()`
- Printing min, max etc of a row: `data.row1.min()` or `data.row1.max`
- Aggregating, Generalising, Summarising -

`data.groupby()`

- Handling missing Values `df.isna()`

  In summary, the transition of data into information usually follows the sequence of data cleansing, modeling \ analysing it, then formating the results so that they can be plotted or shown in table format. pandas accomodates all these requirements.

Note: Please refer link [5] for adetailed insight on pandas functionality.

Analysis of Iris Dataset - Background

Iris dataset was introduced by Ronald Fischer, a british statistician and biologist in 1936. Fischer developed a linear model to distingush the species from each other.

The dataset contains sets of data for 3 varieties of the Iris flower -

- Sentosa
- Versicolor
- Virginica.

50 data readings were collected for these 3 varieties under the following 4 attributes -

- sepal width and sepal length.
- petal width and sepal length.

Note:

Different pandas functionalities have been used within this section where we analyse iris datset.

Checking pandas version

The cell below shows the version of pandas that we are using -

In [1]:

```
import pandas as pd
pd.__version__
```

Out[1]:

'0.25.1'

Reading the dataset

In [3]:

```
data=pd.read_csv("data/iris.csv")
data
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [4]:

```
print("Total Number of (Rows,Columns) :",data.shape)
```

```
Total Number of (Rows,Columns) : (150, 5)
```

Describing the data

The Cell below shows the Count, Mean, Min, Max, Standard Deviation and Median of the data being used. It also shows the Specie with minimum and maximum values.

In [5]:

```
print("Count \n",data.count(axis=0),"\n")
print("Mean \n",data.mean(axis=0),"\n")
print("Min \n",data.min(axis=0),"\n")
print("Max \n",data.max(axis=0),"\n")
print("Median \n",data.median(axis=0),"\n")
print("Std Dev. \n",data.std(axis=0),"\n")
```

```
Count
 sepal_length    150
sepal_width     150
petal_length    150
petal_width     150
species         150
dtype: int64

Mean
 sepal_length    5.843333
sepal_width     3.054000
petal_length    3.758667
petal_width     1.198667
dtype: float64

Min
 sepal_length       4.3
sepal_width          2
petal_length         1
petal_width        0.1
species         setosa
dtype: object

Max
 sepal_length         7.9
sepal_width          4.4
petal_length         6.9
petal_width          2.5
species         virginica
dtype: object

Median
 sepal_length    5.80
sepal_width     3.00
petal_length    4.35
petal_width     1.30
dtype: float64

Std Dev.
 sepal_length    0.828066
sepal_width     0.433594
petal_length    1.764420
petal_width     0.763161
dtype: float64
```

But the cell above does not show the quartile ranges. In order to fetch that detail we can use the describe function mention below. The describe function gives all other details as well along with the quartile ranges. so it is a better thing to use describe().

In [6]:

```
data.describe()
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Using the group by function

The output shows the count, median per type of species. The combination of groupby() + describe() give a more granular look to the values per species.

In [7]:

```
data.groupby("species").count()
```

Out[7]:

	sepal_length	sepal_width	petal_length	petal_width
species				
setosa	50	50	50	50
versicolor	50	50	50	50
virginica	50	50	50	50

In [8]:

```
data.groupby("species").median()
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width
species				
setosa	5.0	3.4	1.50	0.2
versicolor	5.9	2.8	4.35	1.3
virginica	6.5	3.0	5.55	2.0

In [9]:

```
data.groupby("species").describe()
```

Out[9]:

	sepal_length								sepal_width		...	petal_
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%
species												
setosa	50.0	5.006	0.352490	4.3	4.800	5.0	5.2	5.8	50.0	3.418	...	1.575
versicolor	50.0	5.936	0.516171	4.9	5.600	5.9	6.3	7.0	50.0	2.770	...	4.600
virginica	50.0	6.588	0.635880	4.9	6.225	6.5	6.9	7.9	50.0	2.974	...	5.875

3 rows × 32 columns

Visualising the data

We will use three ways to visualise the data here.

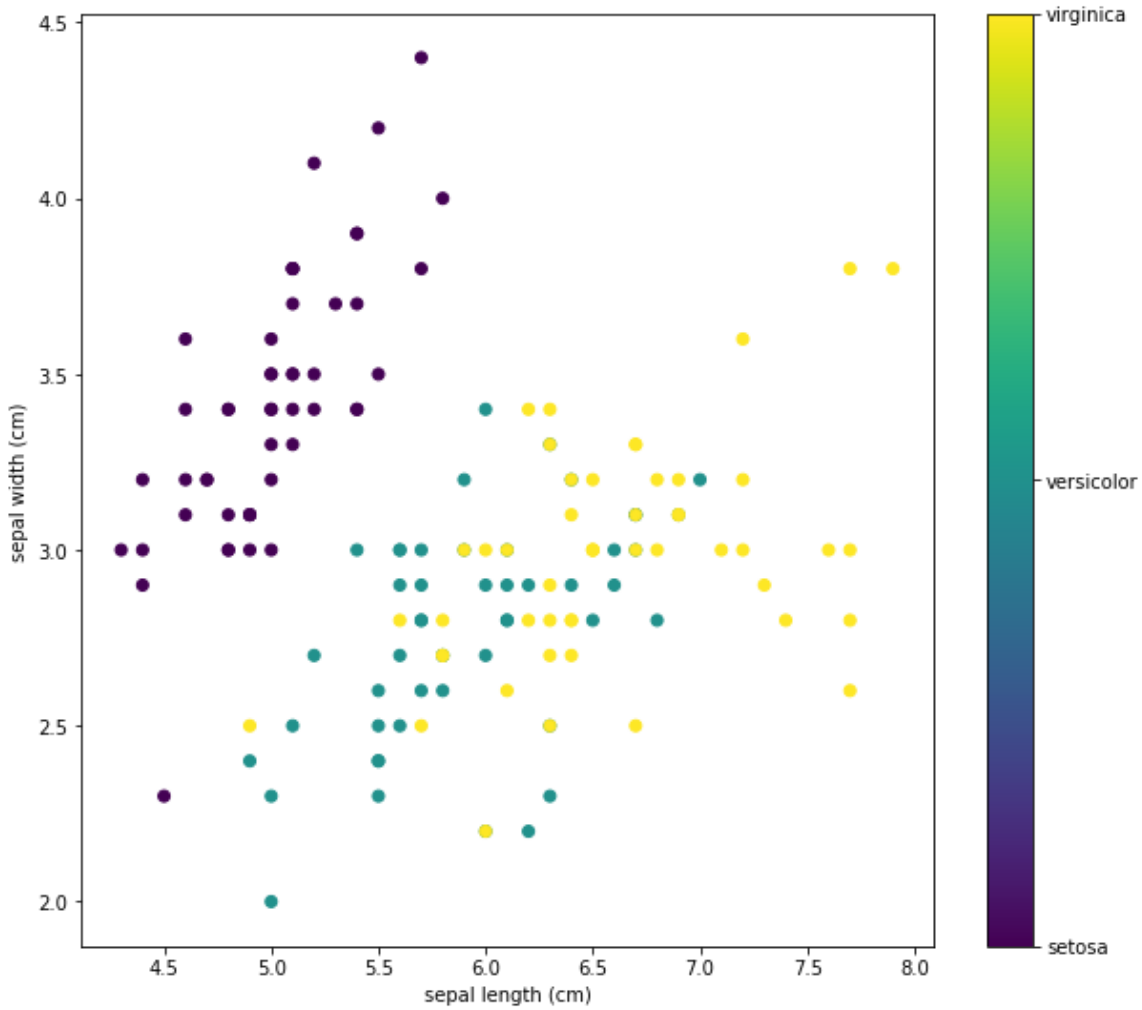
1. Scatter Plot
2. Box plot
3. Histogram
4. Violinplots on each attribute for each species

1. Scatter Plot

Scatter graphs in Python can be created using Matplotlib. Need to import the Iris dataset in a different format than used above for the boxplot. sklearn is a library from the Scikitlearn package where the dataset is stored there.

In [1]:

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
# iris_sklearn
iris_sklearn = load_iris()

# Need to set x axis to 0, i.e.similar to row wise setting in Python3 as used above.

x_index = 0

y_index = 1

# this formatter will label the colorbar with the correct target names
formatter = plt.FuncFormatter(lambda i, *args: iris_sklearn.target_names[int(i)])
# Reference : https://www.scipy-lectures.org/packages/scikit-learn/auto_examples/plot_iris_scatter.html

# Use of 'figsize' command to increase the size of the plot.
plt.figure(figsize=(10, 9))

plt.scatter(iris_sklearn.data[:, x_index], iris_sklearn.data[:, y_index], c=iris_sklearn.target)

plt.colorbar(ticks=[0, 1, 2], format=formatter)

plt.xlabel(iris_sklearn.feature_names[x_index])

plt.ylabel(iris_sklearn.feature_names[y_index])


plt.show()
```



Analysis of Result [7]

Colour has been introduced to distinguish the 3 species from each other.

A scatterplot graphs is used to see trends in the data and is interpreted from left to right.

Advantage of scatterplot:

All data points are visible and the outliers are easier to locate,for example species Setosa(in purple above) has an outlier towards the lower limit and Virginica(in yellow) has 3 outliers towards the upper limit.

Disadvantage:

Where the data shows not follow a linear trend, it is difficult to dicipher relatinships between the variables. It is also very hard to see the exact results for median, quartiles and percentiles.

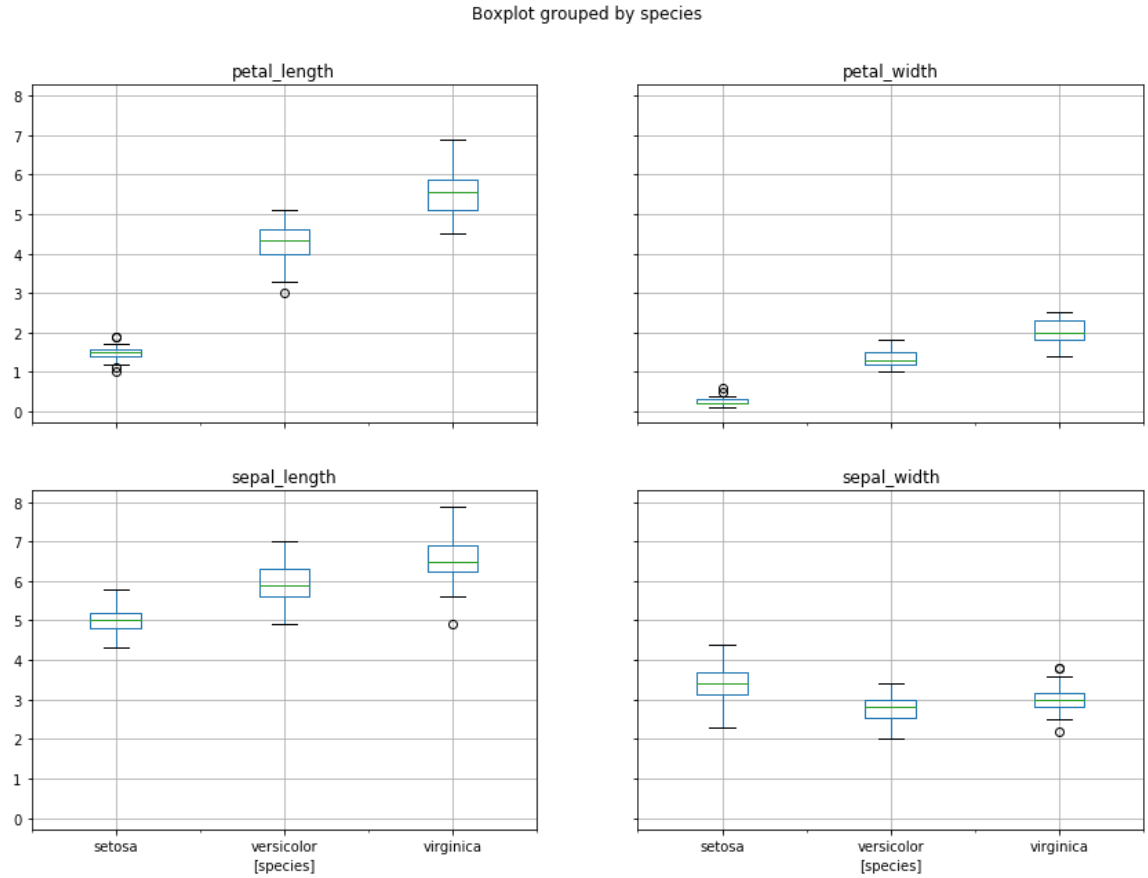
Misinterpretation of results can occur as the satterplot displays relationships, correlations which may not depending on the sample and population sizes.

2. Boxplot [8]

Use of BoxPlot function from Pandas to create the boxplots for each of 3 Species and show the 4 attributes (sepal & petal width and length) Boxplot does not display the plots automatically so need to use the function ".show" from the Matploblib Adding the function "figsize=(12,8)" increases the size of the boxplots as the first output was hard to read

In [11]:

```
import matplotlib.pyplot as plt
data.boxplot(by='species',figsize=(14,10))
plt.show()
```



Analysis of results

Petal length and width:

Setosa is the vaiety with the lowest petal length and width. Virginica the species with the highest Overall, the box plot summary shows that petal length is larger compared to petal width.

Sepal length and width :

Sestosa is the variety with the lowest sepal length but Versicolor has the lowest width. Virginica is the species with the highest sepal length but Sestosa has the highest sepal width. Overall, the box plot summary shows that sepal length is more than sepal width.

Results for individual species:

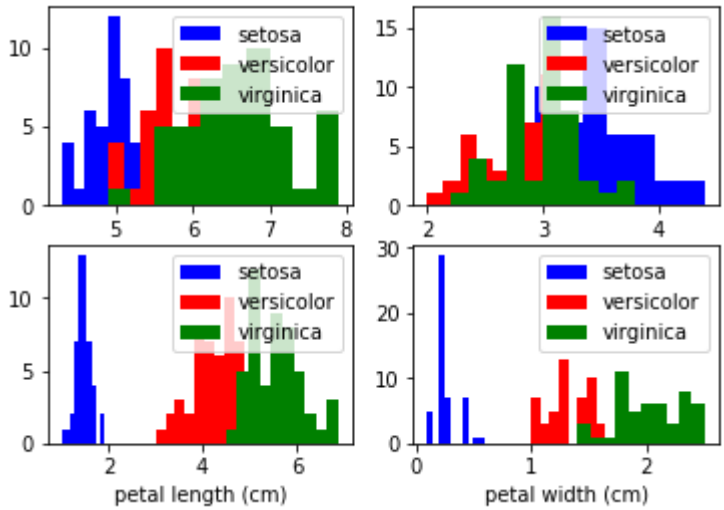
For sepal analysis : Virginica has 3 outliers, 1 each for sepal length and 2 for sepal width. For petal analysis : Sestosa has 5 outliers - 3 for petal length and 2 for petal width. : Versicolor has 1 outlier on the lower extreme.

Histograms compared as the second alternative. [9]

A histogram is used to display the 'underlying frequency distribution (shape) of a set of continuous data.' This concept was first introduced by Karl Pearson

In [12]:

```
import matplotlib.pyplot as plt
# from sklearn.datasets import load_iris

fig, axes = plt.subplots(nrows= 2, ncols=2)
# Assigning different colours to the 3 species, Setosa, Versicolor and Virginica with
colors= ['blue', 'red', 'green']

for i, ax in enumerate(axes.flat):
    for label, color in zip(range(len(iris_sklearn.target_names)), colors):
        ax.hist(iris_sklearn.data[iris_sklearn.target==label, i], label= iris_sklearn.t
arget_names[label], color=color)
        ax.set_xlabel(iris_sklearn.feature_names[i])
        ax.legend(loc='upper right')

plt.figure(figsize=(20,20))
plt.show()
```




```
<Figure size 1440x1440 with 0 Axes>
```

Analysis of results

Similar to scatterplots, colour is also introduced in histograms to distinguish the 3 species from each other. By looking at the histograms initially, petal length and petal width for Sestosa appear to branch away from the normal distribution.

Advantage of histogram:

Histogram makes interpretation of the data much easier. For example, over all the data for sepal width follows a normal distribution. It is easier to identify overlap of the data distribution. In the histogram shown for sepal length, Versicolor adn Virginica species overlap.

Disadvantage of histogram:

Histograms do not display outlier data and therefore misinterpretation of results can occur as there are are known outliers from the boxplot and scatterplot.

Violin Plot

In [13]:

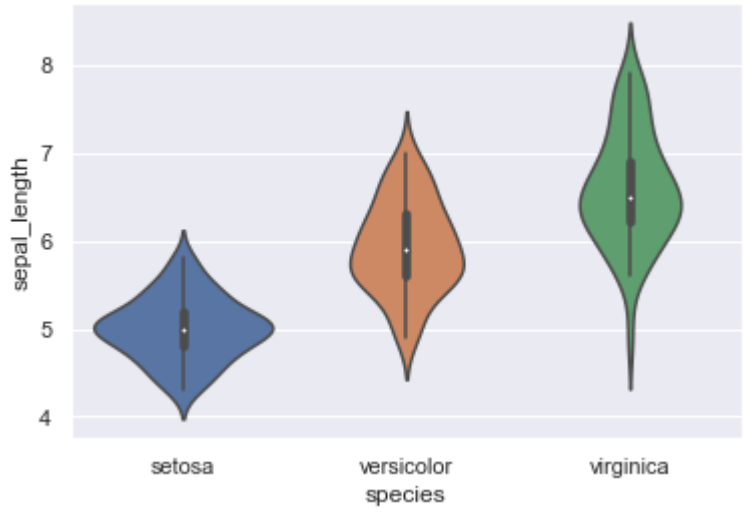
```
import seaborn as sns
sns.set(color_codes=True)
```

In [14]:

```
sns.violinplot(data=data,x="species", y="sepal_length")
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11c9c45f8>
```

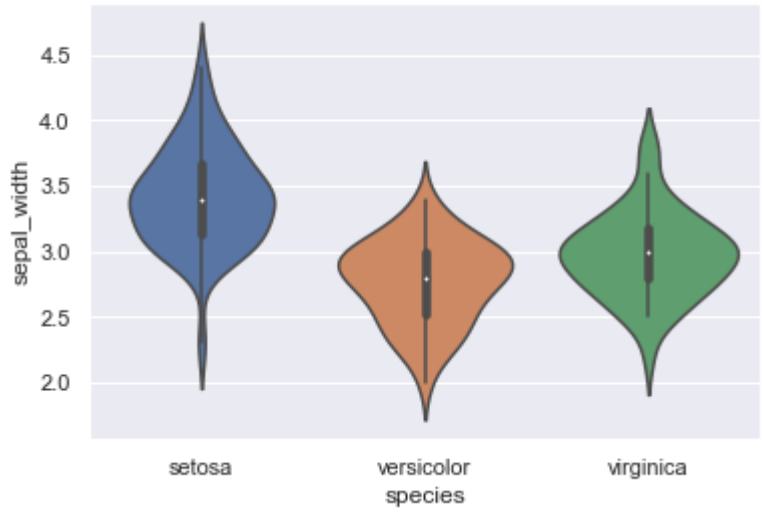


In [15]:

```
sns.violinplot(data=data,x="species", y="sepal_width")
```

Out[15]:

<matplotlib.axes._subplots.AxesSubplot at 0x131ae4fd0>

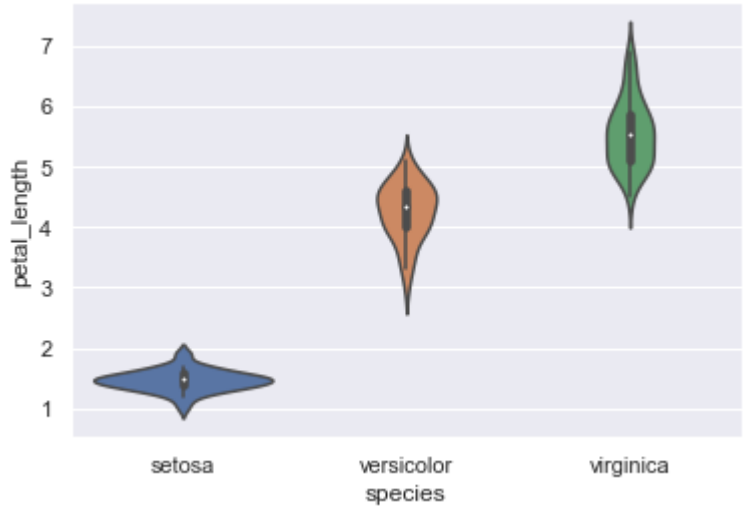


In [16]:

```
sns.violinplot(data=data,x="species", y="petal_length")
```

Out[16]:

<matplotlib.axes._subplots.AxesSubplot at 0x131a9e128>

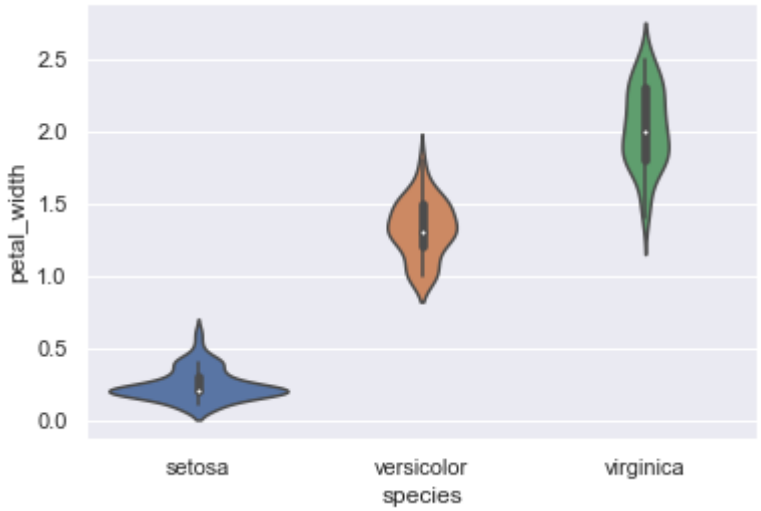


In [17]:

```
sns.violinplot(data=data,x="species", y="petal_width")
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x131cedcc0>
```



Analysis of Results [10] [11]

- The above Violin Boxplot of Species shows that Iris Virginica has highest median value in petal length, petal width and sepal length when compared against Versicolor and Setosa.
- However, Iris Setosa has the highest sepal width median value.
- There is a significant difference between Setosa’s sepal length and sepal width against its petal length and petalwidth.
- That differene is smaller in Versicolor and Virginica.
- The violin plot also indicates that the weight of the Virginica sepal width and petal width are highly concentrated around the median.

Reference links:

[1] https://towardsdatascience.com/be-a-more-efficient-data-scientist-today-master-pandas-with-this-guide-ea362d27386 (https://towardsdatascience.com/be-a-more-efficient-data-scientist-today-master-pandas-with-this-guide-ea362d27386) https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html (https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html)

[2] https://anaconda.org/anaconda/pandas (https://anaconda.org/anaconda/pandas)

[3] https://docs.scipy.org/doc/numpy-1.13.0/user/whatisnumpy.html (https://docs.scipy.org/doc/numpy-1.13.0/user/whatisnumpy.html)

[4] https://towardsdatascience.com/why-and-how-to-use-pandas-with-large-data-9594dda2ea4c (https://towardsdatascience.com/why-and-how-to-use-pandas-with-large-data-9594dda2ea4c)

[5] https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html (https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html)

[6] https://pandas.pydata.org/pandas-docs/stable/whatsnew/v0.25.0.html (https://pandas.pydata.org/pandas-docs/stable/whatsnew/v0.25.0.html)

[7] https://www.dummies.com/education/math/statistics/how-to-interpret-a-scatterplot/ (https://www.dummies.com/education/math/statistics/how-to-interpret-a-scatterplot/)

[8] https://pandas.pydata.org/pandas (https://pandas.pydata.org/pandas)

[9] https://statistics.laerd.com/statistical-guides/understanding-histograms.php (https://statistics.laerd.com/statistical-guides/understanding-histograms.php)

[10] https://www.kaggle.com/adityabhat24/iris-data-analysis-and-machine-learning-python (https://www.kaggle.com/adityabhat24/iris-data-analysis-and-machine-learning-python)

[11] http://rstudio-pubs-static.s3.amazonaws.com/321676_20be34434fe44ed2b229eadeabe0eb98.html (http://rstudio-pubs-static.s3.amazonaws.com/321676_20be34434fe44ed2b229eadeabe0eb98.html)