APSSDC

Andhra Pradesh State Skill Development Corporation

(https://apssdc.in)

Day09 Data Analysis Using Python

Data Visualization using Seaborn

Visualization Packages

Data Visualization using Seaborn

- Seaborn is a Python data visualization library based on matplotlib
- It provides a high-level interface for drawing attractive and informative statistical graphics
- Seaborn is a library for making statistical graphics in Python
- Applications:
  - used in visualising data in Machine learning, data Science
  - statistical aggregation to produce informative plots

Day12 Objectives

- Using Seaborn Styles
- Categorical scatterplots:
  - stripplot() (with kind="strip"; the default)
  - swarmplot() (with kind="swarm")
- Categorical distribution plots:
  - boxplot() (with kind="box")
  - violinplot() (with kind="violin")
- Joint plot
- Regression Plots
- Creating heatmaps
- Creating pairplots

1. Darkgrid
2. Whitegrid
3. Dark
4. White
5. Ticks

Iris Flower DataSet

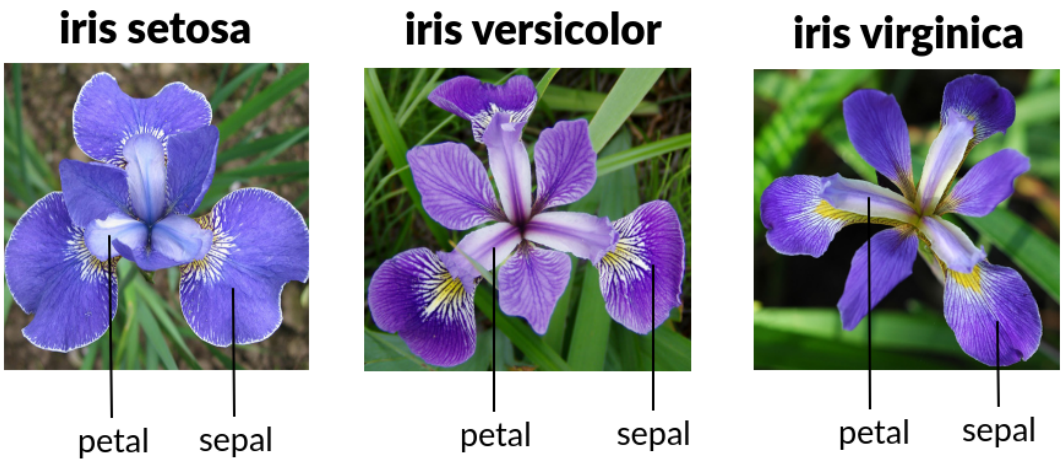
Toy Dataset: Iris Dataset: [https://en.wikipedia.org/wiki/Iris_flower_data_set] (https://en.wikipedia.org/wiki/Iris_flower_data_set%5D)

- A simple dataset to learn the basics.
- 3 flowers of Iris species. [see images on wikipedia link above]
- 1936 by Ronald Fisher.
- Petal and Sepal: http://terpconnect.umd.edu/~petersd/666/html/iris_with_labels.jpg (http://terpconnect.umd.edu/~petersd/666/html/iris_with_labels.jpg) # * Objective: Classify a new flower as belonging to one of the 3 classes given the 4 features.
- Importance of domain knowledge.
- Why use petal and sepal dimensions as features?
- Why do we not use 'color' as a feature?

The four features of these 3 types of flowers (Iris setosa, Iris versicolor, Iris virginica) are:

1. Sepal Length
2. Sepal Width
3. Petal Length
4. Petal Width

We will classify the given flower by using this Four Features



In [2]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

In [3]:
```
1 sns.get_dataset_names()
```

Out[3]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'exercise',
 'flights',
 'fmri',
 'gammas',
 'geyser',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'tips',
 'titanic']
```

In [4]:
```
1 iris = sns.load_dataset('iris')
```

In [5]:
```
1 iris.head()
```

Out[5]:

	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

In [6]:
```
1 iris.shape
```

Out[6]:
```
(150, 5)
```

In [7]:
```
1 iris['species'].value_counts()
```

Out[7]:
```
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

In [8]:
```
1 iris.groupby('species').count()
```

Out[8]:

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

In [9]:
```
1 iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [10]:
```
1 iris.describe()
```

Out[10]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
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

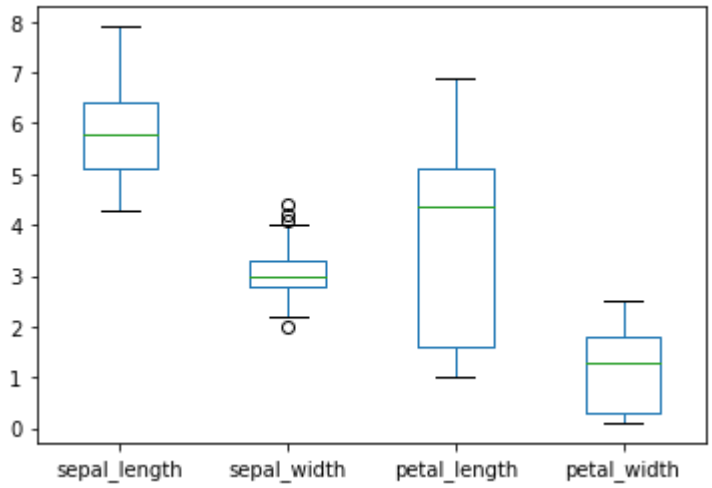
In [12]:
```
1 iris.columns
```

Out[12]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [ ]:
```
1
```

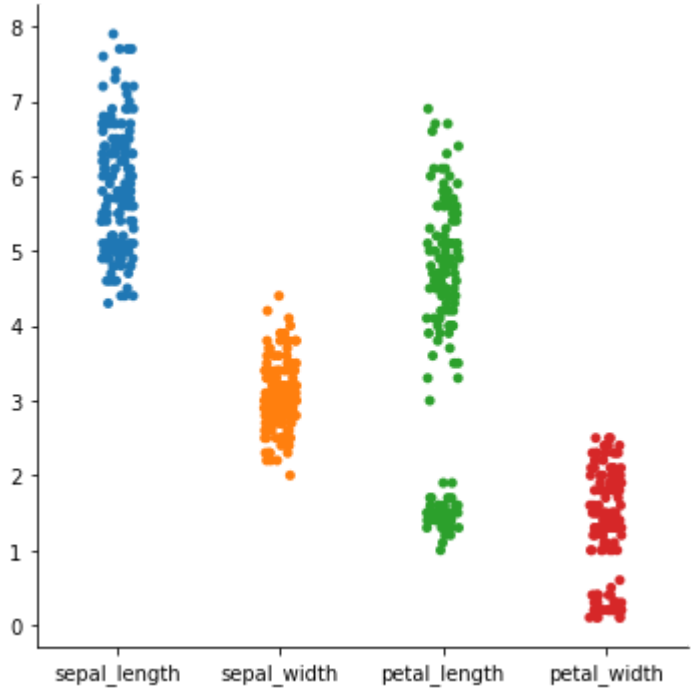
In [16]:
```
1 iris.plot(kind = 'box')
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x2498ce86700>



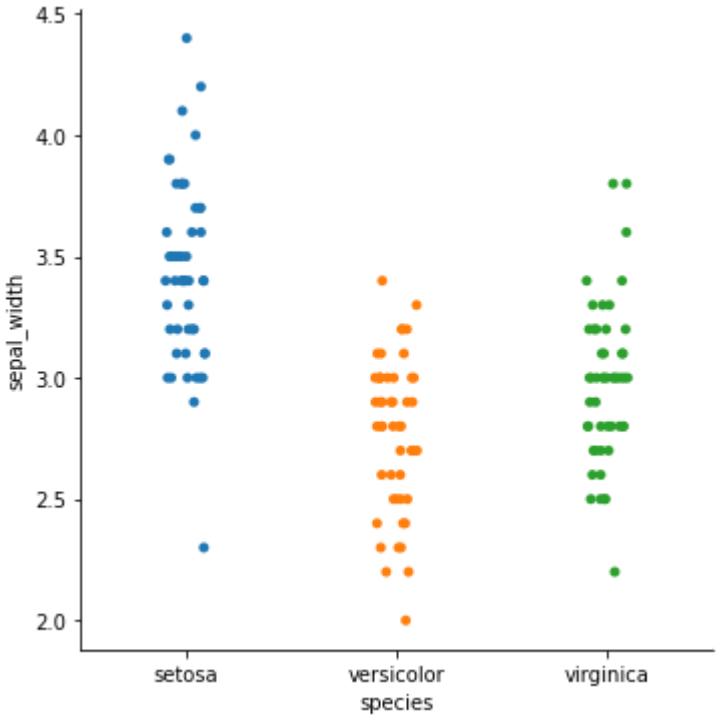
In [21]:
```
1 sns.catplot(data = iris)
```

Out[21]: <seaborn.axisgrid.FacetGrid at 0x249fbd68a00>



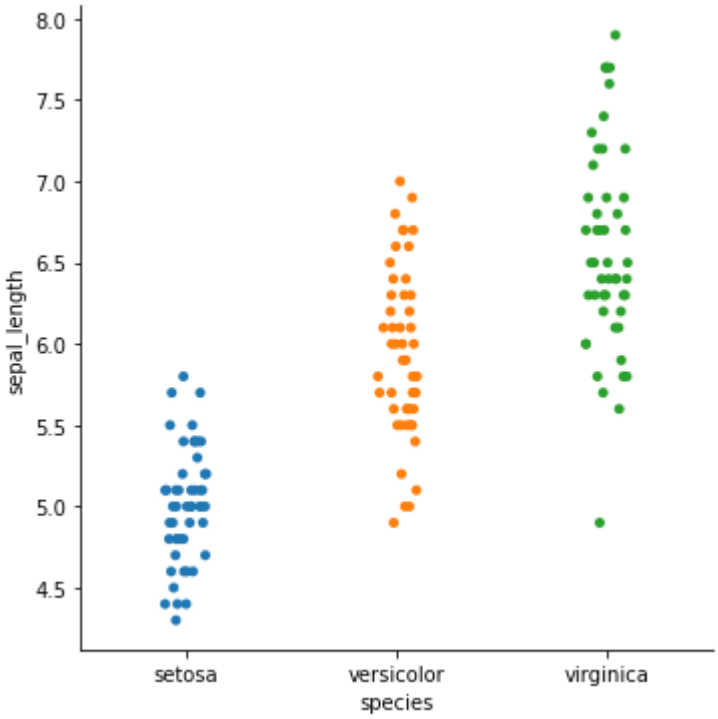
In [23]:
```
1 sns.catplot(x = 'species', y = 'sepal_width', data = iris)
```

Out[23]: <seaborn.axisgrid.FacetGrid at 0x2498f056f70>



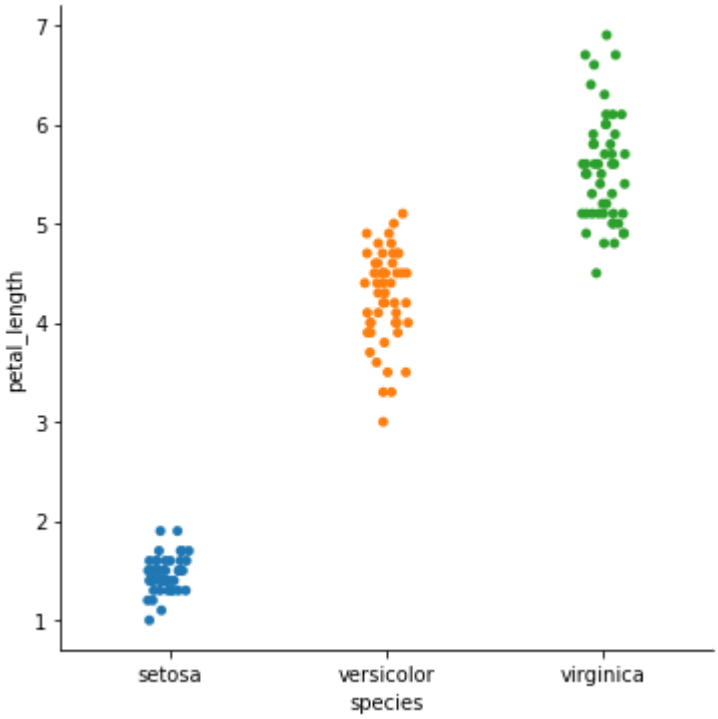
In [24]:
```
1 sns.catplot(x = 'species', y = 'sepal_length', data = iris)
```

Out[24]: <seaborn.axisgrid.FacetGrid at 0x2498f07e520>



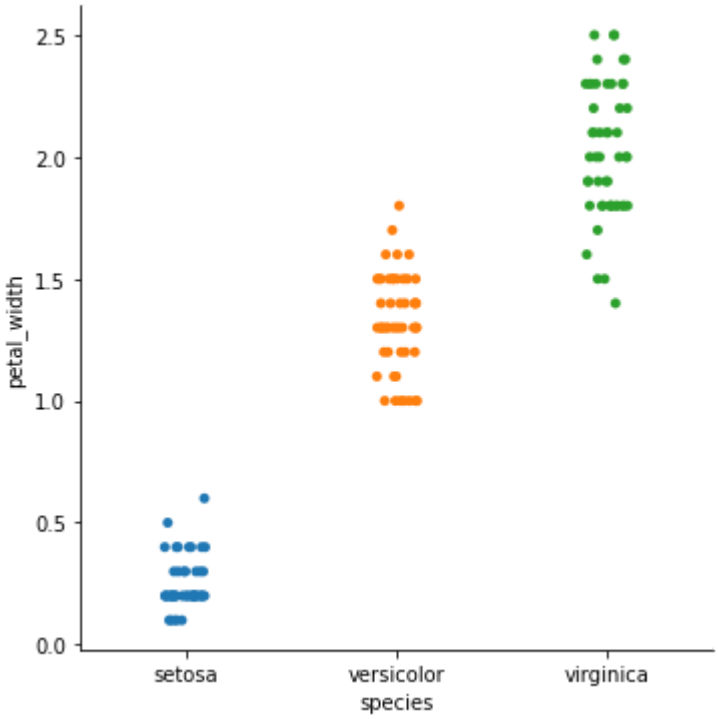
In [25]:
```
1 sns.catplot(x = 'species', y = 'petal_length', data = iris)
```

Out[25]: <seaborn.axisgrid.FacetGrid at 0x2498ef35ee0>



In [26]:
```
1 sns.catplot(x = 'species', y = 'petal_width', data = iris)
```

Out[26]: <seaborn.axisgrid.FacetGrid at 0x2498ef937f0>

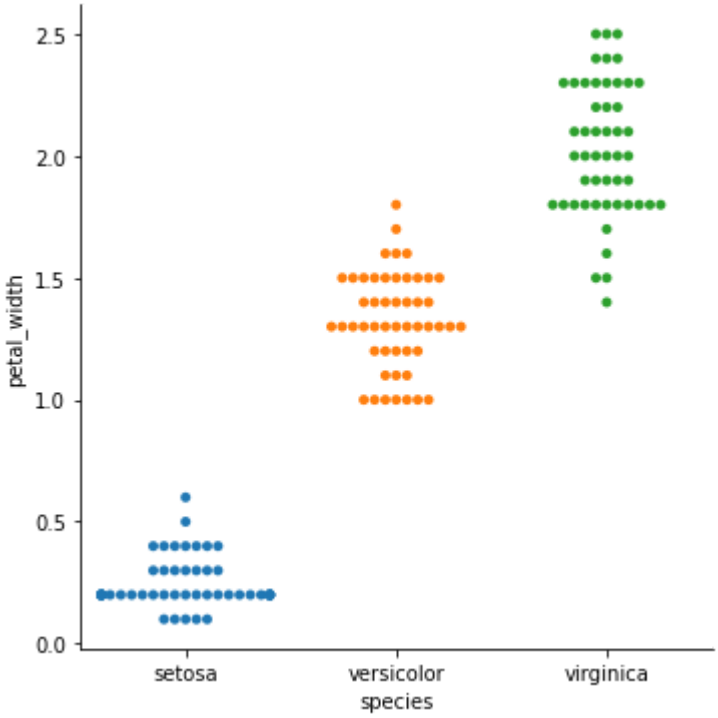


In [27]:

```
1 sns.catplot(x = 'species', y = 'petal_width', data = iris, kind = 'swarm
```

```
C:\Users\Jesus\anaconda3\lib\site-packages\seaborn\categorical.py:1296: Use
rWarning: 28.0% of the points cannot be placed; you may want to decrease th
e size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

Out[27]: <seaborn.axisgrid.FacetGrid at 0x2498f33a2b0>

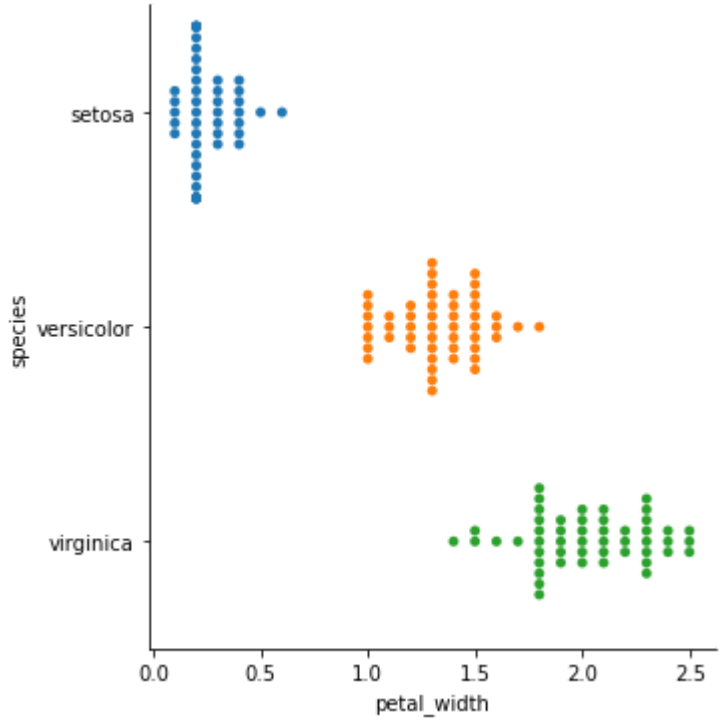


In [28]:

```
1 sns.catplot(y = 'species', x = 'petal_width', data = iris, kind = 'swarm
```

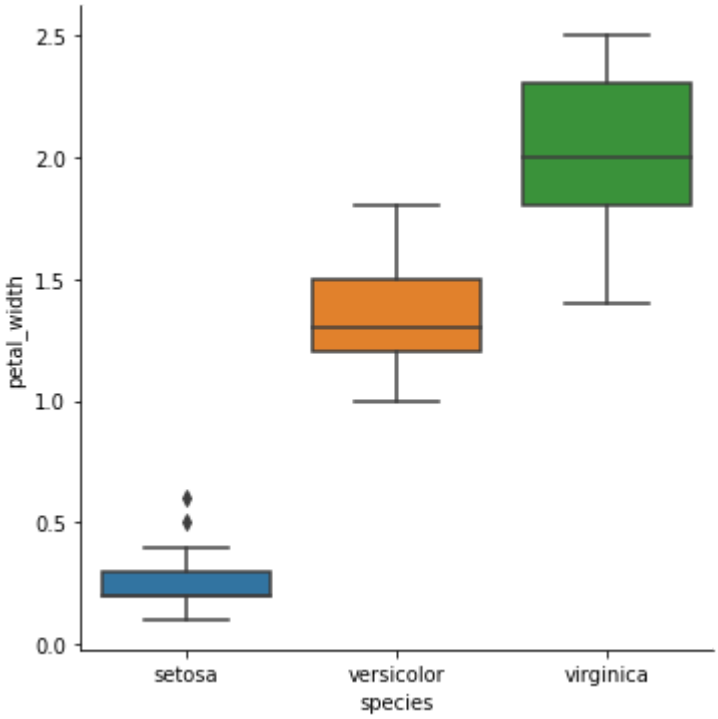
```
C:\Users\Jesus\anaconda3\lib\site-packages\seaborn\categorical.py:1296: Use
rWarning: 24.0% of the points cannot be placed; you may want to decrease th
e size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

Out[28]: <seaborn.axisgrid.FacetGrid at 0x2498f23d220>



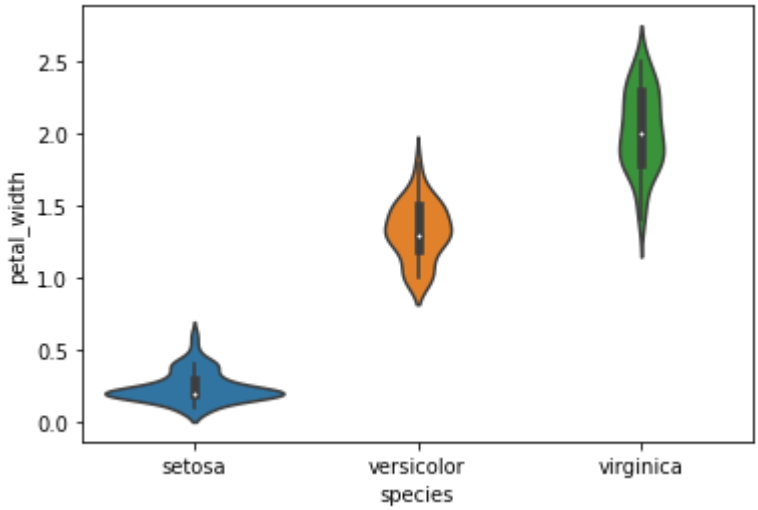
In [29]:
```
1 sns.catplot(x = 'species', y = 'petal_width', data = iris, kind = 'box')
```

Out[29]: <seaborn.axisgrid.FacetGrid at 0x2498f27d700>



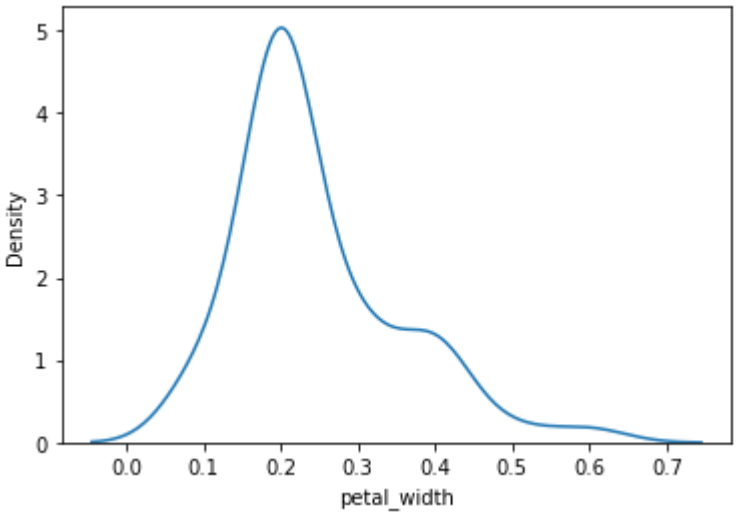
In [33]:
```
1 sns.violinplot(x = 'species', y = 'petal_width', data = iris)
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x249904ac370>



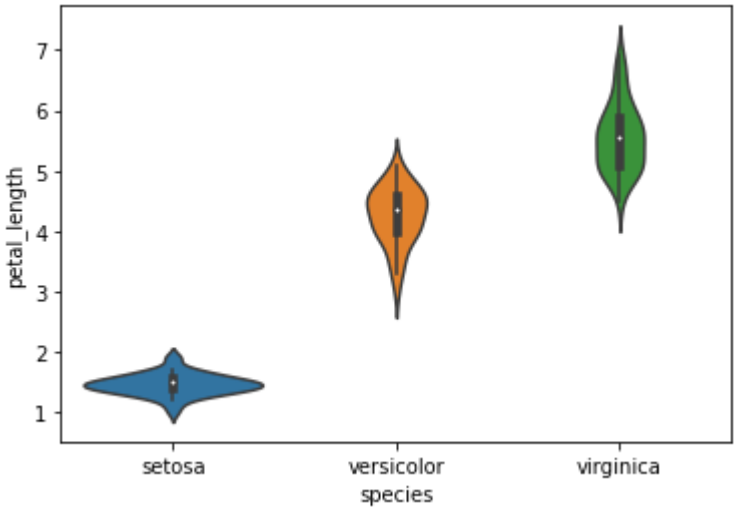
In [36]:
```
1 setosa = iris[iris['species'] == 'setosa']['petal_width']
2
3 sns.kdeplot(x = setosa , data = iris)
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x2499258be20>



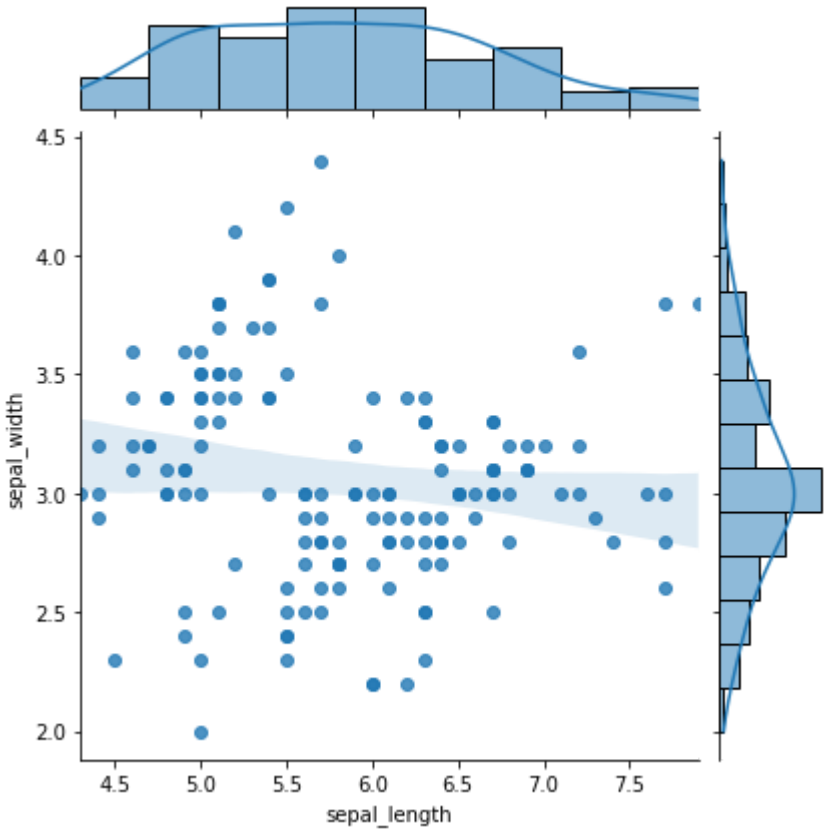
In [37]:
```
1 sns.violinplot(x = 'species', y = 'petal_length', data = iris)
```

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x249925e1dc0>



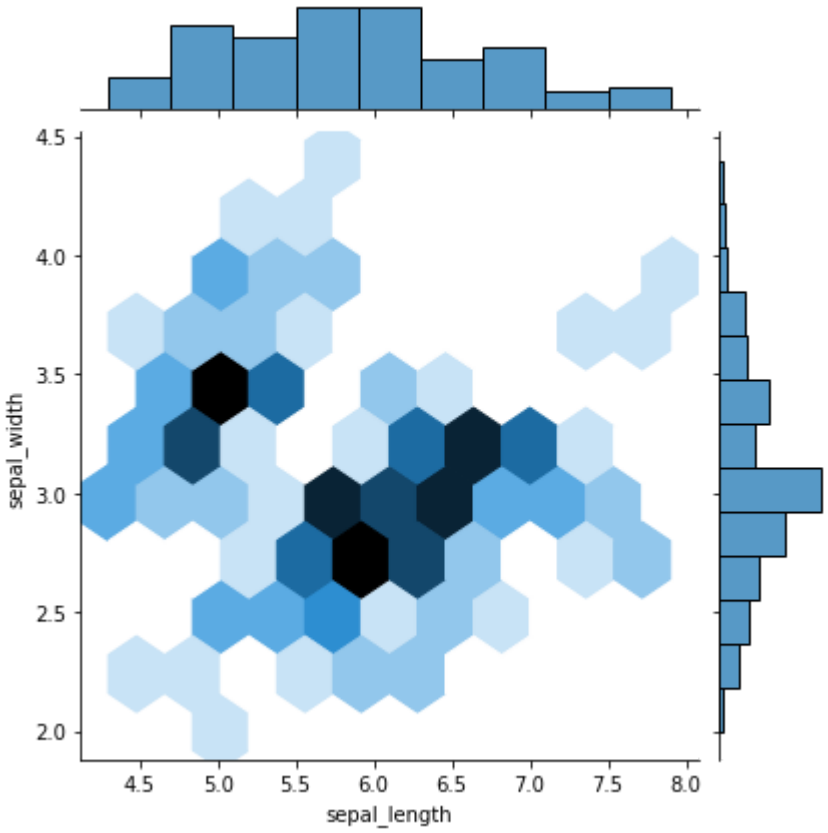
In [38]:
```
1 sns.jointplot(x = 'sepal_length', y = 'sepal_width', data = iris, kind =
```

Out[38]: <seaborn.axisgrid.JointGrid at 0x24992631a30>



In [39]:
```
1 sns.jointplot(x = 'sepal_length', y = 'sepal_width', data = iris, kind =
```

Out[39]: <seaborn.axisgrid.JointGrid at 0x249927686d0>



In [65]:
```
1 sns.pairplot(data = iris, hue = 'species')
```

Out[65]: <seaborn.axisgrid.PairGrid at 0x1d6cba32100>

In [41]:

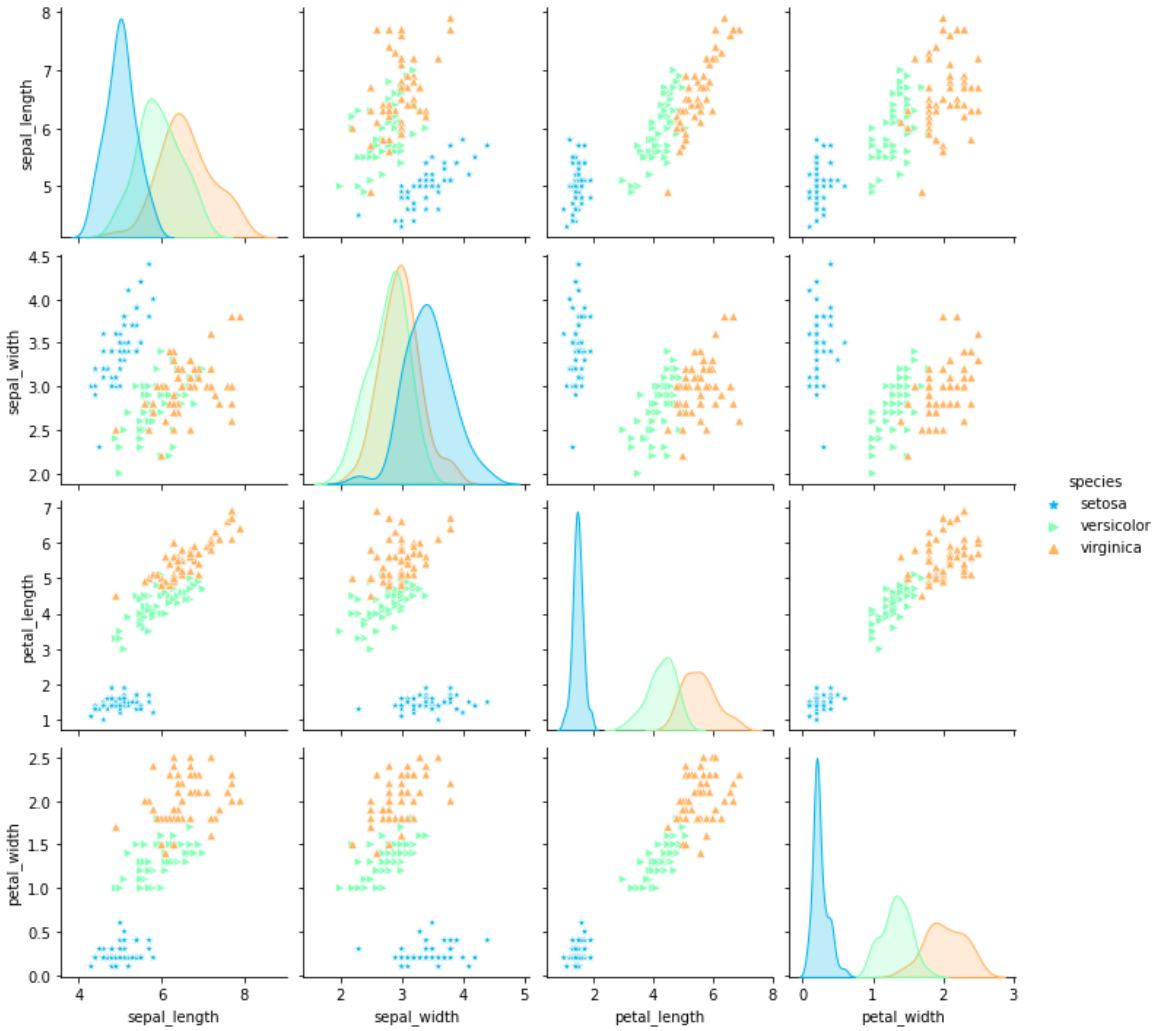
```
help(sns.pairplot)
```

```
    height : scalar
        Height (in inches) of each facet.
    aspect : scalar
        Aspect * height gives the width (in inches) of each facet.
    corner : bool
        If True, don't add axes to the upper (off-diagonal) triangle of t
he
        grid, making this a "corner" plot.
    dropna : boolean
        Drop missing values from the data before plotting.
    {plot, diag, grid}_kws : dicts
        Dictionaries of keyword arguments. ``plot_kws`` are passed to the
        bivariate plotting function, ``diag_kws`` are passed to the univa
riate
        plotting function, and ``grid_kws`` are passed to the :class:`Pai
rGrid`
        constructor.

    Returns
    -------
```

In [42]:
```
sns.pairplot(data = iris, hue = 'species', markers = ['*', '>', '^'], palett
```

Out[42]: <seaborn.axisgrid.PairGrid at 0x249929f4250>



In [43]:
```
corr = iris.corr()
```

In [44]:
```
corr
```

Out[44]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.117570	0.871754	0.817941
sepal_width	-0.117570	1.000000	-0.428440	-0.366126
petal_length	0.871754	-0.428440	1.000000	0.962865
petal_width	0.817941	-0.366126	0.962865	1.000000

In [45]:
```
1 sns.heatmap(corr)
```

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x24994d4f190>

In [46]:
```
1 sns.heatmap(corr, annot = True)
```

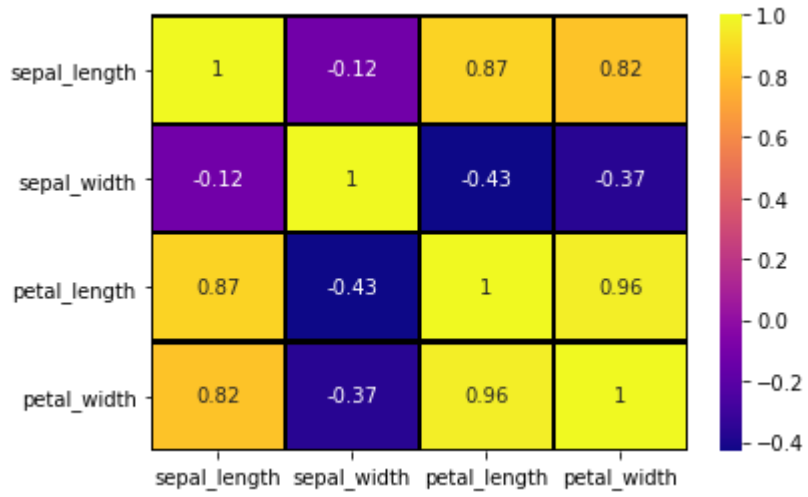
Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x2499525ccd0>

In [47]:
```
help(sns.heatmap)
```

```
        Using this parameter will change the default ``cmap`` if none is
        specified.
    robust : bool, optional
        If True and ``vmin`` or ``vmax`` are absent, the colormap range i
s
        computed with robust quantiles instead of the extreme values.
    annot : bool or rectangular dataset, optional
        If True, write the data value in each cell. If an array-like with
the
        same shape as ``data``, then use this to annotate the heatmap ins
tead
        of the data. Note that DataFrames will match on position, not ind
ex.
    fmt : str, optional
        String formatting code to use when adding annotations.
    annot_kws : dict of key, value mappings, optional
        Keyword arguments for :meth:`matplotlib.axes.Axes.text` when ``an
not``
        is True.
    linewidths : float, optional
```

In [48]:
```
sns.heatmap(corr, annot = True, cmap = 'plasma', linecolor = '#000000',
```

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x24993827280>




```
np.random.randn --> Normal/Gaussian -> mean = 0, variance = std = 1
```

In [49]:

```
1 help(np.random.normal)
```

```
Help on built-in function normal:

normal(...) method of numpy.random.mtrand.RandomState instance
    normal(loc=0.0, scale=1.0, size=None)

    Draw random samples from a normal (Gaussian) distribution.

    The probability density function of the normal distribution, first
    derived by De Moivre and 200 years later by both Gauss and Laplace
    independently [2]_, is often called the bell curve because of
    its characteristic shape (see the example below).

    The normal distributions occurs often in nature.  For example, it
    describes the commonly occurring distribution of samples influenced
    by a large number of tiny, random disturbances, each with its own
    unique distribution [2]_.

    .. note::
        New code should use the ``normal`` method of a ``default_rng()``
```