In [1]:
```
# Seaborn
# Supervised Learning
```

In [2]:
```
pip install seaborn
```

```
Note: you may need to restart the kernel to use updated packages.

'C:\Users\ravi' is not recognized as an internal or external command,
operable program or batch file.
```

In [3]:
```
import seaborn as sns
```

In [4]:
```
sns.get_dataset_names()
```

```
C:\Users\ravi sastry\anaconda3\lib\site-packages\seaborn\utils.py:384: Gues
sedAtParserWarning: No parser was explicitly specified, so I'm using the be
st available HTML parser for this system ("lxml"). This usually isn't a pro
blem, but if you run this code on another system, or in a different virtual
environment, it may use a different parser and behave differently.

The code that caused this warning is on line 384 of the file C:\Users\ravi
sastry\anaconda3\lib\site-packages\seaborn\utils.py. To get rid of this war
ning, pass the additional argument 'features="lxml"' to the BeautifulSoup c
onstructor.

  gh_list = BeautifulSoup(http)
```

Out[4]:
```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'exercise',
 'flights',
 'fmri',
 'gammas',
 'geyser',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'tips',
 'titanic']
```

In [5]:
```
df = sns.load_dataset('iris')
```

In [6]:
```
1 df.head()
```

Out[6]:

	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 [7]:
```
1 df.shape
```

Out[7]: (150, 5)

In [8]:
```
1 df.tail()
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width	species
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

In [10]:
```
1 df.sample(5)
```

Out[10]:

	sepal_length	sepal_width	petal_length	petal_width	species
5	5.4	3.9	1.7	0.4	setosa
56	6.3	3.3	4.7	1.6	versicolor
38	4.4	3.0	1.3	0.2	setosa
86	6.7	3.1	4.7	1.5	versicolor
81	5.5	2.4	3.7	1.0	versicolor

In [11]:
```
1 df['species'].count()
```

Out[11]: 150

In [13]:
```
1 df.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 [14]:
```
1 df['species'].value_counts()
```

Out[14]:
```
virginica     50
versicolor    50
setosa        50
Name: species, dtype: int64
```

In [15]:
```
1 df.groupby("species").count()
```

Out[15]:

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

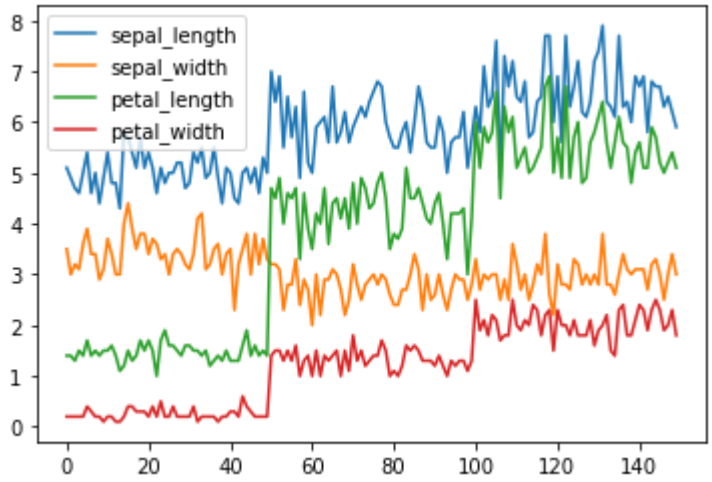
In [16]:
```
1 df.describe()
```

Out[16]:

	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

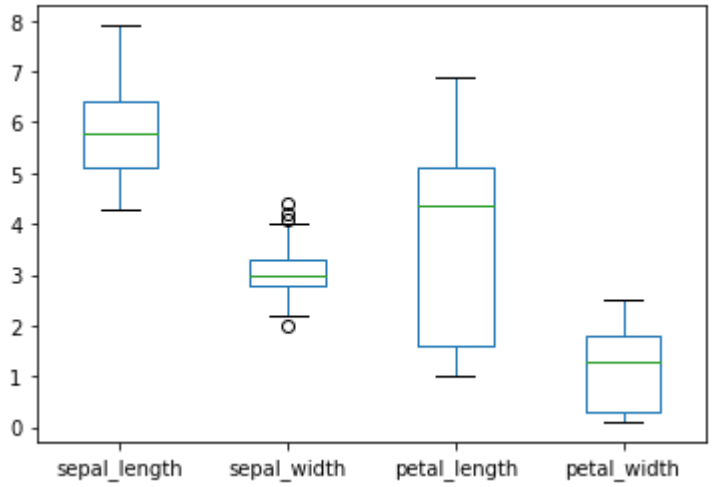
In [17]:
```
1 df.plot()
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x2cd2f812850>



In [18]:
```
1 df.plot(kind='box')
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x2cd2ffa41f0>



In [19]:
```
1 dir(sns)
```

Out[19]:
```
['FacetGrid',
 'JointGrid',
 'PairGrid',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 '__version__',
 '__warningregistry__',
 '_orig_rc_params',
 'algorithms',
 'axes_style',
 'axisgrid',
 'barplot',
 'blend_palette',
 'boxenplot',
 'boxplot',
 'categorical',
 'catplot',
 'choose_colorbrewer_palette',
 'choose_cubehelix_palette',
 'choose_dark_palette',
 'choose_diverging_palette',
 'choose_light_palette',
 'clustermap',
 'cm',
 'color_palette',
 'colors',
 'countplot',
 'crayon_palette',
 'crayons',
 'cubehelix_palette',
 'dark_palette',
 'desaturate',
 'despine',
 'distplot',
 'distributions',
 'diverging_palette',
 'dogplot',
 'external',
 'factorplot',
 'get_data_home',
 'get_dataset_names',
 'heatmap',
 'hls_palette',
 'husl_palette',
 'jointplot',
 'kdeplot',
 'light_palette',
```

```
'lineplot',
'lmplot',
'load_dataset',
'lvplot',
'matrix',
'miscplot',
'mpl',
'mpl_palette',
'pairplot',
'palettes',
'palplot',
'plotting_context',
'pointplot',
'rcmod',
'regplot',
'regression',
'relational',
'relplot',
'reset_defaults',
'reset_orig',
'residplot',
'rugplot',
'saturate',
'scatterplot',
'set',
'set_color_codes',
'set_context',
'set_hls_values',
'set_palette',
'set_style',
'stripplot',
'swarmplot',
'utils',
'violinplot',
'widgets',
'xkcd_palette',
'xkcd_rgb']
```

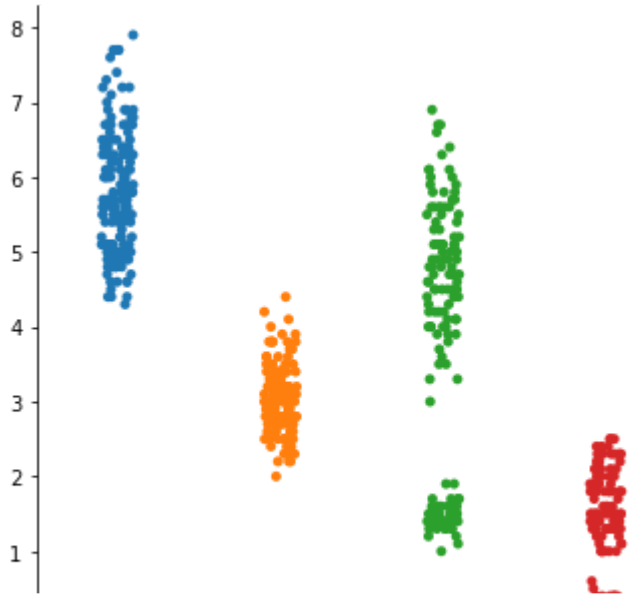
In [20]:
```
1 df.head()
```

Out[20]:

	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

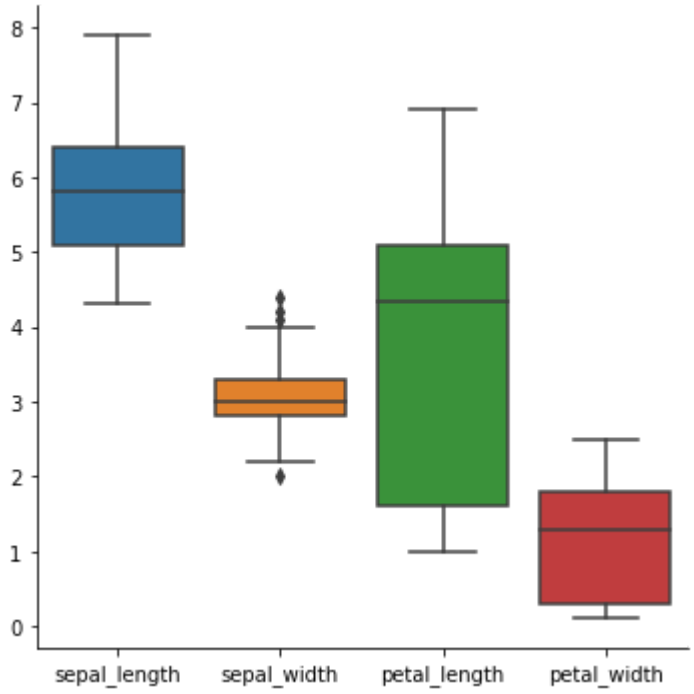
In [22]:
```
1 ##catplot
2 sns.catplot(data=df)
```

Out[22]: <seaborn.axisgrid.FacetGrid at 0x2cd3004c6d0>



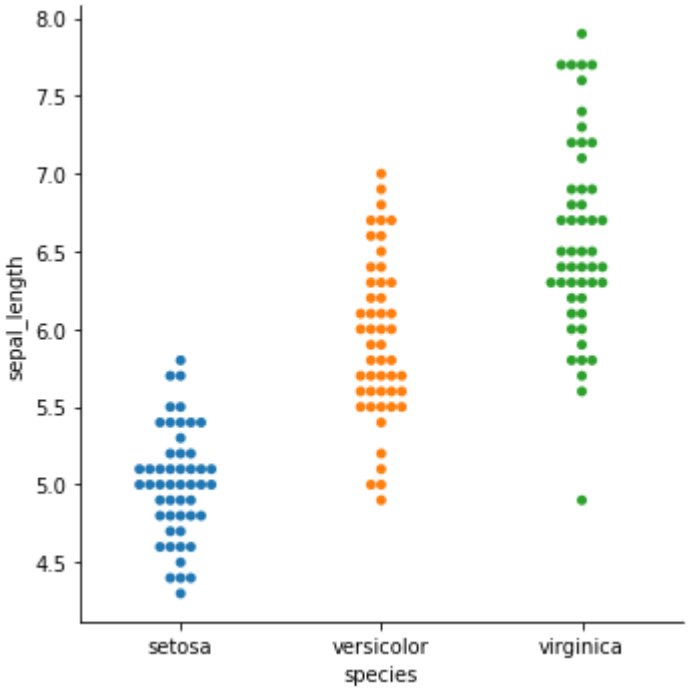
In [23]:
```
1 sns.catplot(data=df,kind='box')
```

Out[23]: <seaborn.axisgrid.FacetGrid at 0x2cd300bb160>



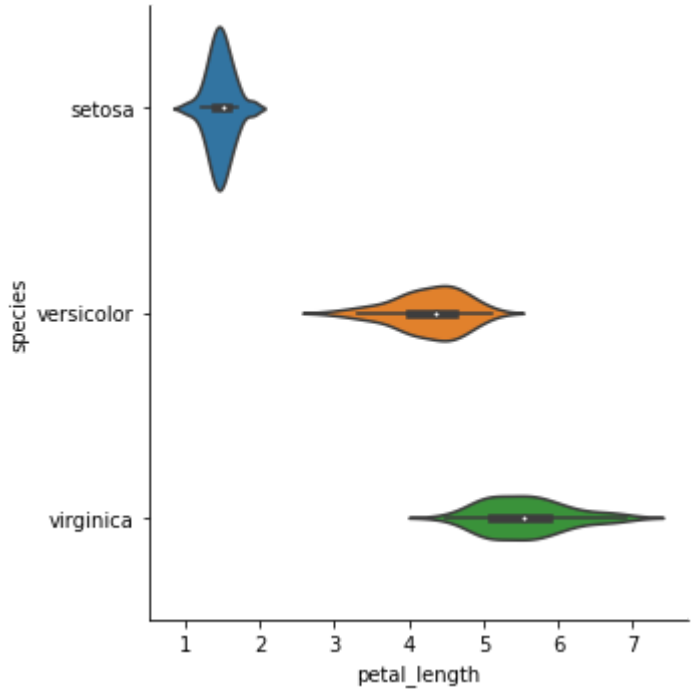
In [24]:
```
1 sns.catplot(x="species",y='sepal_length',data=df,hue='species',kind='swa
```

Out[24]: <seaborn.axisgrid.FacetGrid at 0x2cd300bb910>



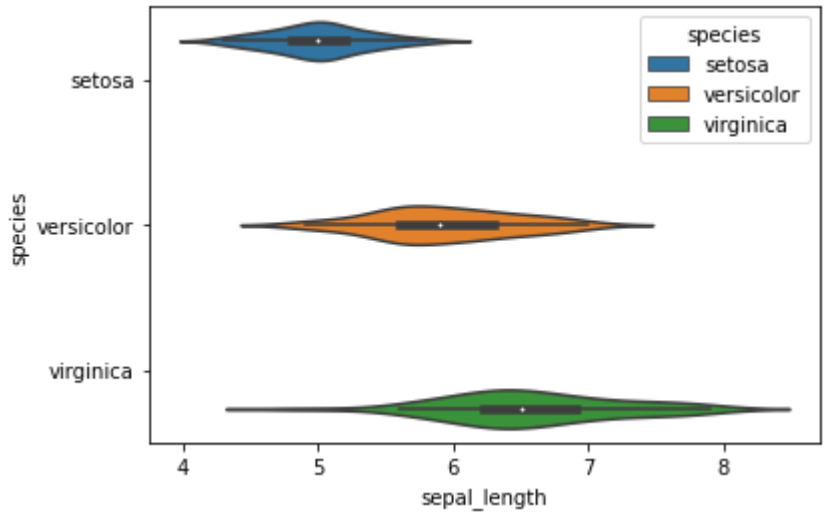
In [26]:
```
1 sns.catplot(y='species',x="petal_length",data=df,kind="violin")
```

Out[26]: <seaborn.axisgrid.FacetGrid at 0x2cd3045d820>



In [28]:
```
1 sns.violinplot(x='sepal_length',y="species",data=df,hue='species')
```

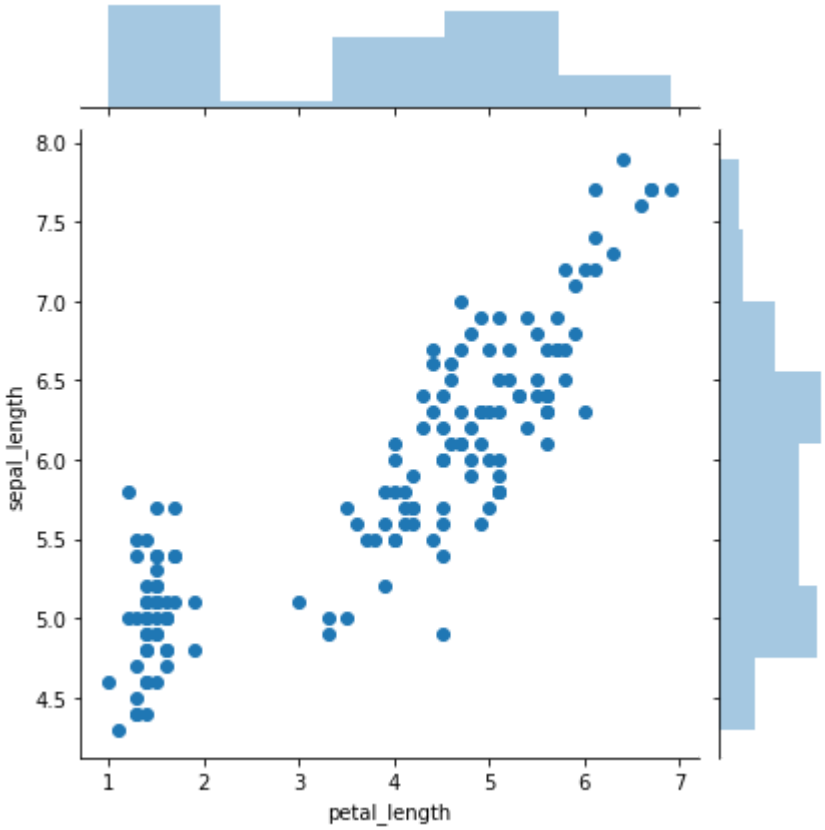
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x2cd2f5f7c40>



In [30]:
```
1 # kde plot
2 ### kernal density estimator plot
3 ##### its uses distribution data
4 #sns.kdeplot(y='sepal_length',data=df)
```

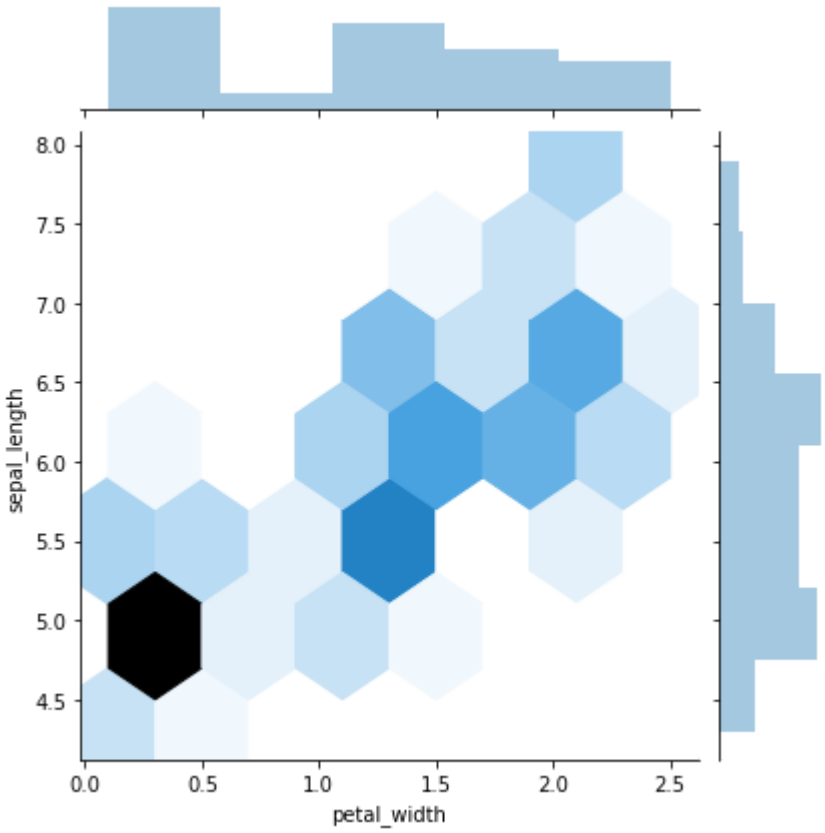
In [31]:
```
# joint plot
### joint the different plots together

sns.jointplot(y='sepal_length',x='petal_length',data=df)
```

Out[31]: <seaborn.axisgrid.JointGrid at 0x2cd30812fd0>



In [35]:
```
1 sns.jointplot(y='sepal_length',x="petal_width",data=df,kind='hex')
```

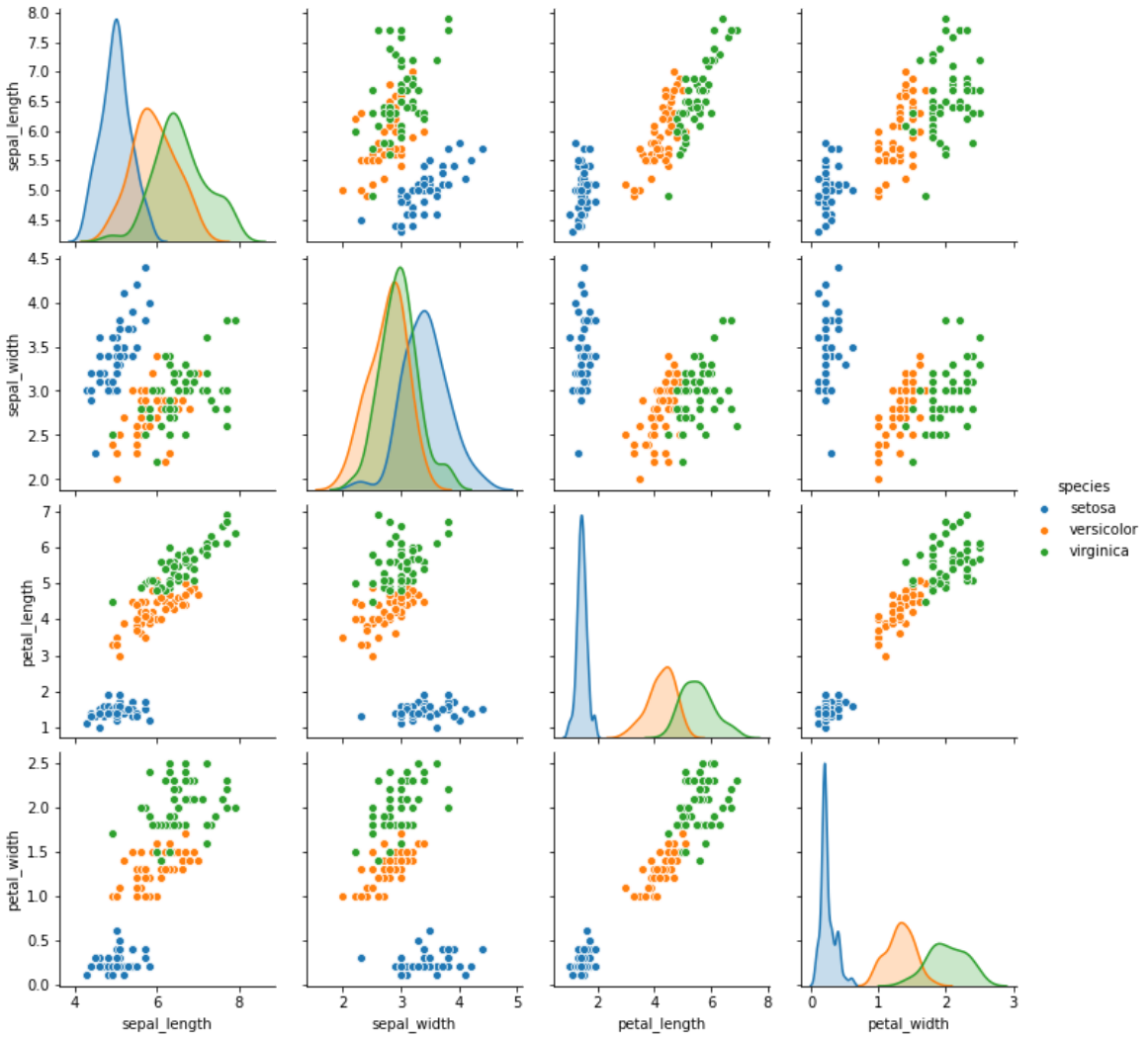
Out[35]: <seaborn.axisgrid.JointGrid at 0x2cd31bfdf40>



In [36]:
```
1 # pairplot
```

In [40]:
```
1 sns.pairplot(data=df,hue="species")
```

Out[40]: <seaborn.axisgrid.PairGrid at 0x2cd32283940>



In [41]:
```
1 corr=df.corr()
```

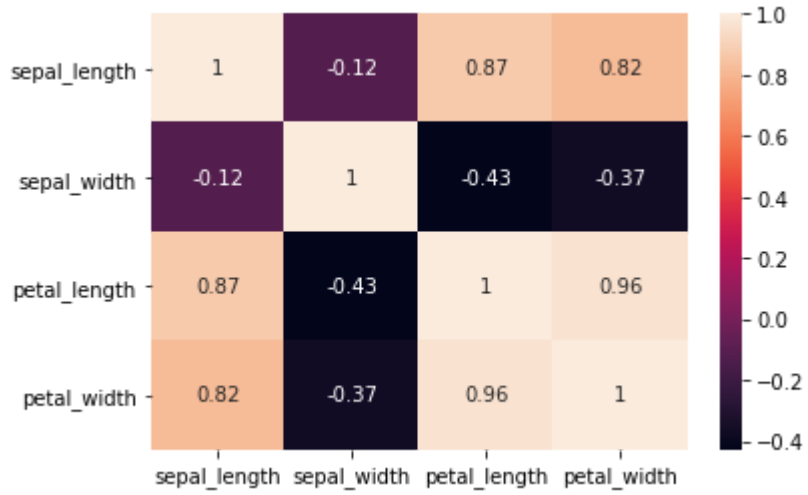
In [42]:
```
1 corr
```

Out[42]:

	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

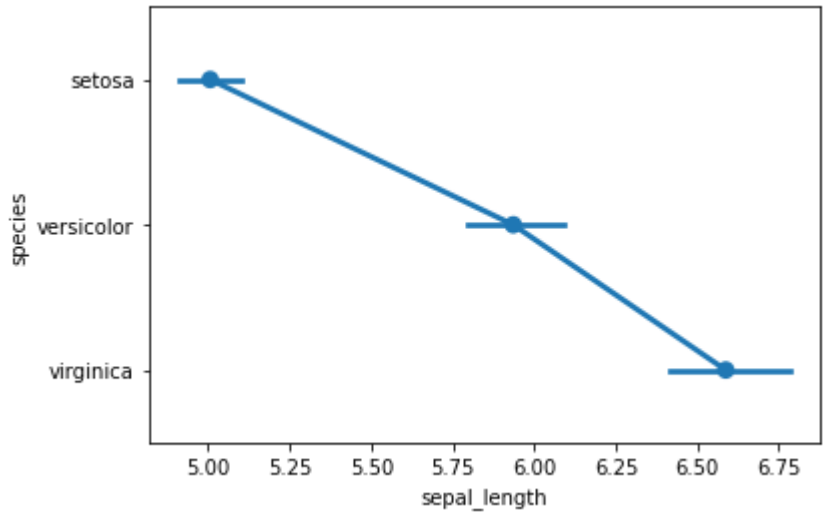
In [47]:
```
1 sns.heatmap(corr,annot=True)
```

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x2cd34fb1c70>



In [48]:
```
1 sns.pointplot(x='sepal_length',y='species',data=df)
```

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x2cd3507d3d0>



Supervised Learning

- Regression
  - dependence on continuos data
    - Linear Regression
      - linear regression with one feature
      - linear regression with multiple features
    - polynomial Regression
      - PR with one feature
      - PR with Multiple features
- Classification
  - dependes on categorical data(0,1,True,False,Yes,No

- Knn - K-Nearest Neibhours classifier/regressor
- Logistic Regression
- SVM - support vector Machine
- Decision Tree Classifier/Regressor
- Random Forest regressor/classifier

Linear Regression with One feature

- Linear model is sum weighted predicted data to target values
- Y=mx+c
- y = target value/OUtput
- m = slope
- x = Input Value
- c = Coefficient/Intercept

Slope Formula

- m= (x-xmean)*(y-ymean)/(x-xmean)^2

Coeffient or intercept formula

- c= ymean-(m*xmean)

ML steps:

- 1.Get or load the data
- 2.Preprocessing the data
- 3.Define input and output
- 4.Apply the model or algorithm
- 5.pass the data to training and testing
- 6.calculate the score or accuracy score

In [49]:

```
# get the libraries
import pandas as pd
import numpy as np

```

In [50]:
```
# 1.get the data
df = pd.read_csv("https://raw.githubusercontent.com/AP-State-Skill-Devel
df.head()
```

Out[50]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

In [51]:
```
# 2.preprocessing the data
df.shape
```

Out[51]: (30, 2)

In [52]:
```
df.isnull().sum()
```

Out[52]:
```
YearsExperience    0
Salary             0
dtype: int64
```

In [53]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

In [54]:
```
df.describe()
```

Out[54]:

	YearsExperience	Salary
count	30.000000	30.000000
mean	5.313333	76003.000000
std	2.837888	27414.429785
min	1.100000	37731.000000
25%	3.200000	56720.750000
50%	4.700000	65237.000000
75%	7.700000	100544.750000
max	10.500000	122391.000000

In [55]:
```
df.columns
```

Out[55]:
```
Index(['YearsExperience', 'Salary'], dtype='object')
```

In [56]:
```
## 3. Define input and output
x=df[['YearsExperience']]
y=df['Salary']
```

In [57]:
```
## 4.Apply the model or algorithm
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x,y)
```

Out[57]:
```
LinearRegression()
```

In [58]:
```
## 5.test the data
model.predict([[7.1],[10]])
```

Out[58]:
```
array([ 92886.932681  , 120291.82341322])
```

In [59]:
```
1 df
```

Out[59]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

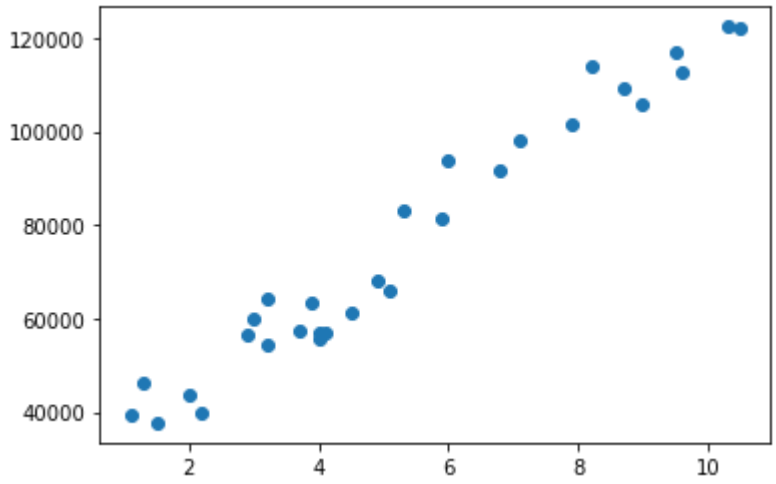
In [60]:
```
# 6.identify the score
model.score(x,y)*100
```

Out[60]: 95.69566641435085

In [61]:
```
# visualize the relationship of input and output
import matplotlib.pyplot as plt
```

In [62]:
```
plt.scatter(df['YearsExperience'],df['Salary'],label = 'actual data')
```

Out[62]: <matplotlib.collections.PathCollection at 0x2cd352aa6d0>



In [ ]:
```

```