Introduction

Data visualization is the graphic representation of data. It involves producing images that communicate relationships among the represented data to viewers. Visualizing data is an essential part of data analysis and machine learning. We'll use Python libraries *Matplotlib* and *Seaborn* to learn and apply some popular data visualization techniques. We'll use the words chart, plot, and graph interchangeably in this tutorial.

In [125]:

```
# Importing Libraries

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

Line Plot

The line chart is one of the simplest and most widely used data visualization techniques. A line chart displays information as a series of data points or markers connected by straight lines. You can customize the shape, size, color, and other aesthetic elements of the lines and markers for better visual clarity.

In [96]:

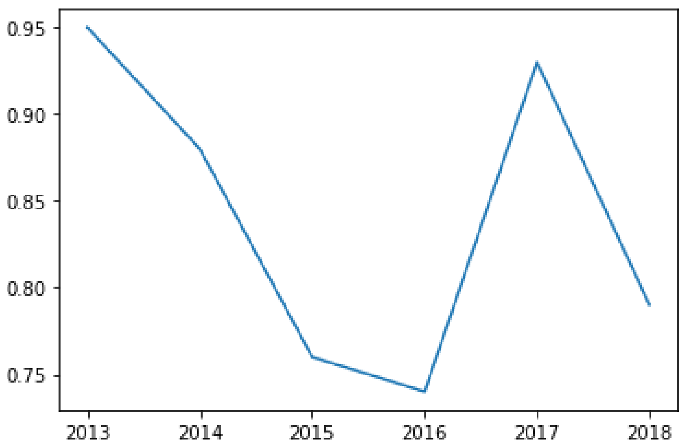
```
yield_oranges = [0.95, 0.88, 0.76, 0.74, 0.93, 0.79]
years = [2013, 2014, 2015, 2016, 2017, 2018]
```

In [97]:

```
plt.plot(years, yield_oranges)
```

Out[97]:

```
[<matplotlib.lines.Line2D at 0x1d6784ed088>]
```

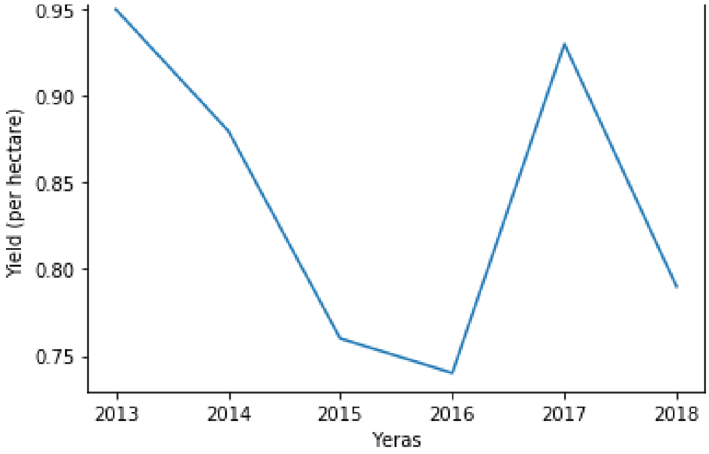


In [98]:

```
plt.plot(years, yield_oranges)
plt.xlabel('Yeras')
plt.ylabel('Yield (per hectare)')
```

Out[98]:

```
Text(0, 0.5, 'Yield (per hectare)')
```



In [99]:

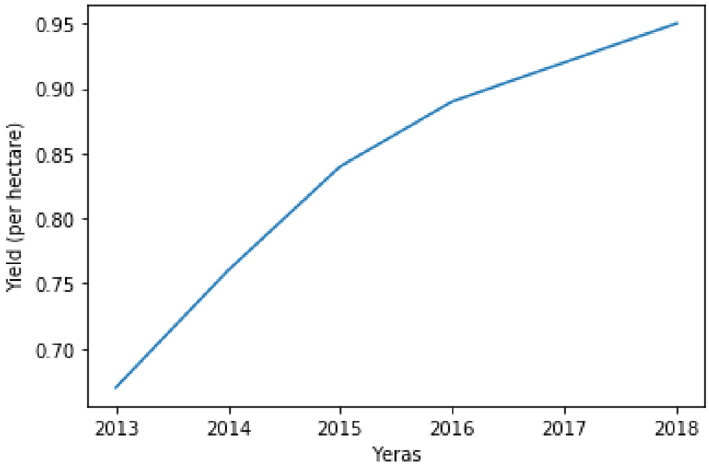
```
yield_apples = [0.67, 0.76, 0.84, 0.89, 0.92, 0.95]
years = [2013, 2014, 2015, 2016, 2017, 2018]
```

In [100]:

```
plt.plot(years, yield_apples)
plt.xlabel('Yeras')
plt.ylabel('Yield (per hectare)')
```

Out[100]:

```
Text(0, 0.5, 'Yield (per hectare)')
```

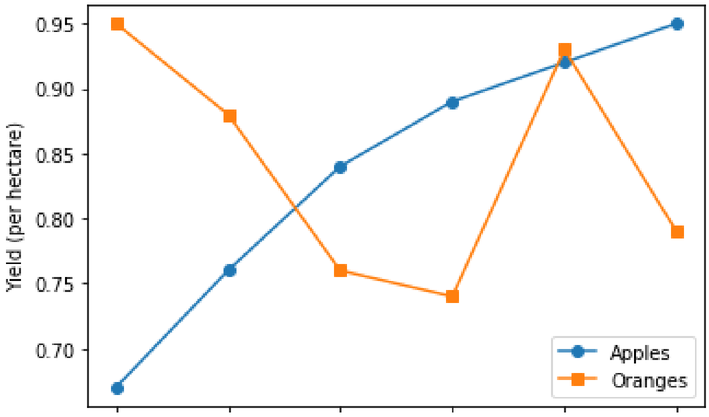


In [101]:

```
plt.plot(years, yield_apples, marker='o')
plt.plot(years, yield_oranges, marker='s')
plt.xlabel('Yeras')
plt.ylabel('Yield (per hectare)')
plt.legend(['Apples', 'Oranges'])
```

Out[101]:

```
<matplotlib.legend.Legend at 0x1d6787b1bc8>
```



In [102]:

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

In [103]:

```
data.head()
```

Out[103]:

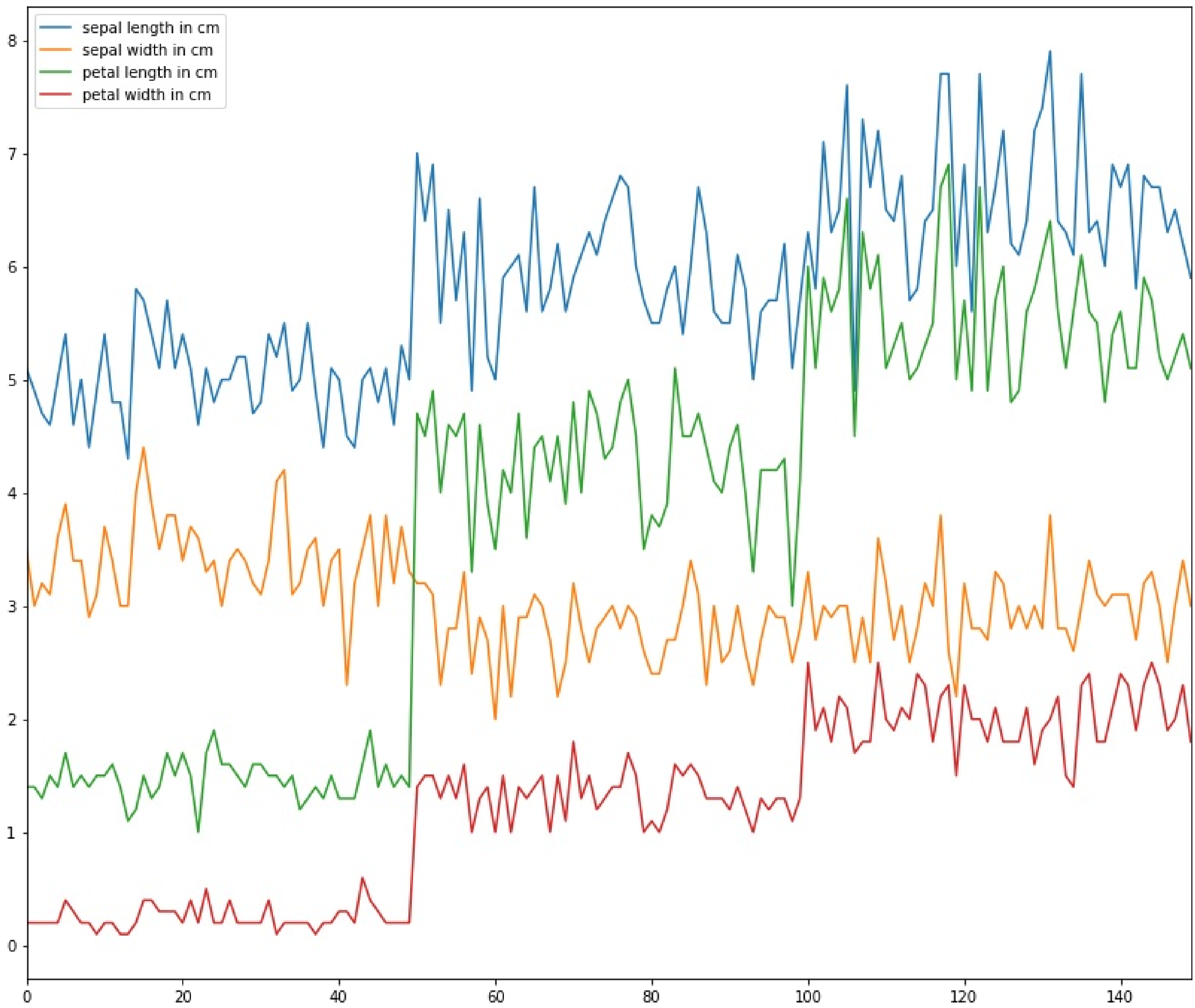
	sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	outputs
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [104]:

```
data.plot(figsize=(14,12))
```

Out[104]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d6787f1a08>
```



Scatter Plot

In a scatter plot, the values of 2 variables are plotted as points on a 2-dimensional grid. Additionally, you can also use a third variable to determine the size or color of the points. Let's try out an example.

In [105]:

```
data
```

Out[105]:

	sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	outputs
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [106]:

```
data['outputs'].unique()
```

Out[106]:

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [107]:

```
data['outputs'].value_counts()
```

Out[107]:

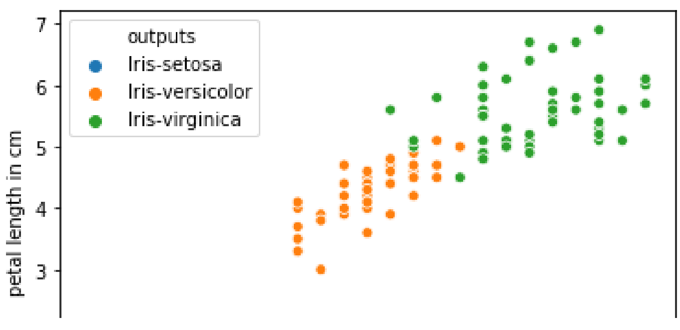
```
Iris-virginica     50
Iris-versicolor    50
Iris-setosa        50
Name: outputs, dtype: int64
```

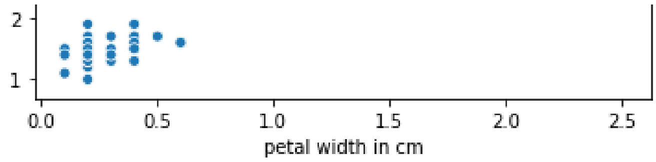
In [108]:

```
sns.scatterplot(x = data['petal width in cm'], y = data['petal length in cm'], hue=data[
'outputs'])
```

Out[108]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d678867c88>
```





In [109]:

```
X = data.drop('outputs', axis = 1)
```

In [110]:

```
X
```

Out[110]:

	sepal length in cm	sepal width in cm	petal length in cm	petal width in cm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [111]:

```
data
```

Out[111]:

	sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	outputs
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [112]:

```
data['outputs']
```

Out[112]:

```
0          Iris-setosa
1          Iris-setosa
2          Iris-setosa
3          Iris-setosa
4          Iris-setosa
              ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: outputs, Length: 150, dtype: object
```

PairPlot

In [115]:

```
# Pairwise scatter plot: Pairplot

sns.set_style("whitegrid")
sns.pairplot(data, hue = 'outputs', size = 3)
plt.show()
```

Histogram

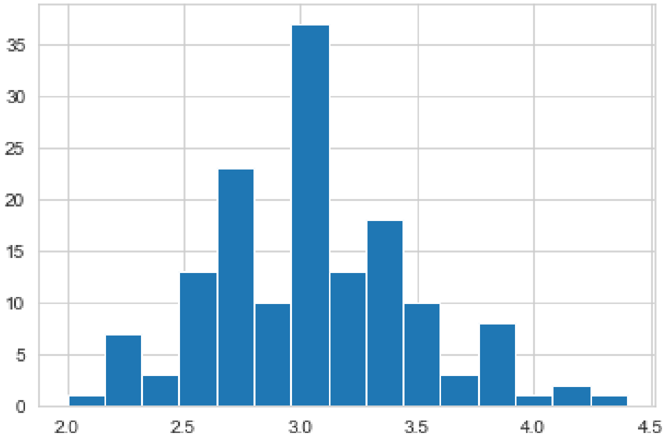
A histogram represents the distribution of a variable by creating bins (interval) along the range of values and showing vertical bars to indicate the number of observations in each bin.

In [116]:

```
plt.hist(data['sepal width in cm'], bins = 15)
```

Out[116]:

```
(array([ 1.,  7.,  3., 13., 23., 10., 37., 13., 18., 10.,  3.,  8.,  1.,
         2.,  1.]),
 array([2.  , 2.16, 2.32, 2.48, 2.64, 2.8 , 2.96, 3.12, 3.28, 3.44, 3.6 ,
        3.76, 3.92, 4.08, 4.24, 4.4 ]),
 <a list of 15 Patch objects>)
```



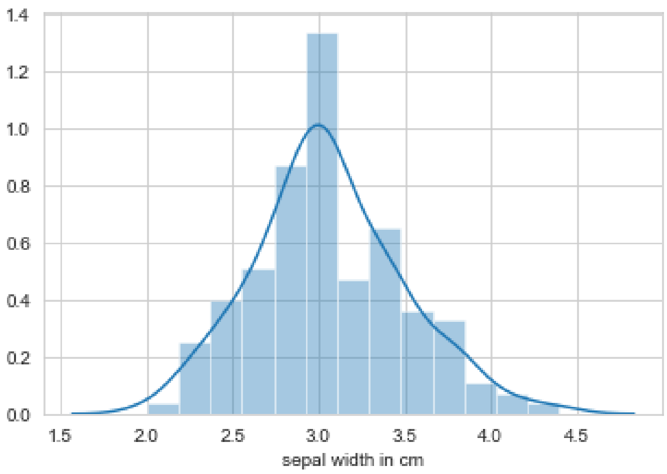
Dist Plot

In [117]:

```
sns.distplot(data['sepal width in cm'])
```

Out[117]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d67a84fec8>
```

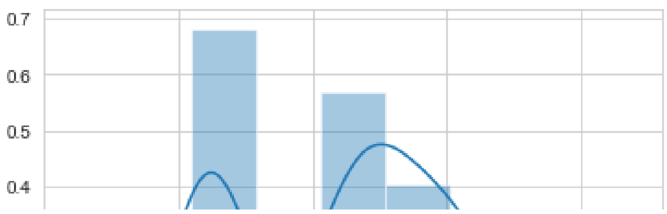


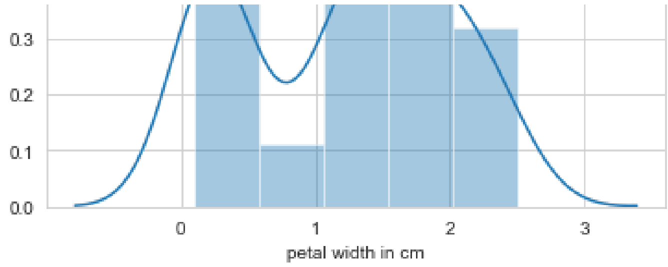
In [118]:

```
sns.distplot(data['petal width in cm'])
```

Out[118]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d67a8dad88>
```





Box Plot

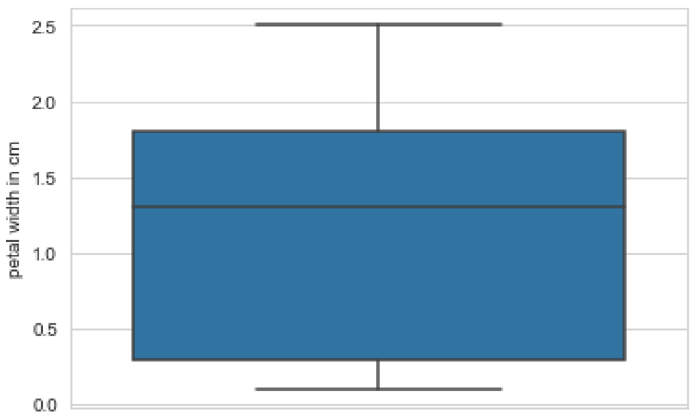
A boxplot shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution, except the points that are determined to be *outliers* using a method that is a function of the inter-quartile range.

In [119]:

```
sns.boxplot(data['petal width in cm'], orient = 'v')
```

Out[119]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d67a913688>
```



In [120]:

```
train = pd.read_csv("titanic - Copy.csv")
```

In [121]:

```
train.head()
```

Out[121]:

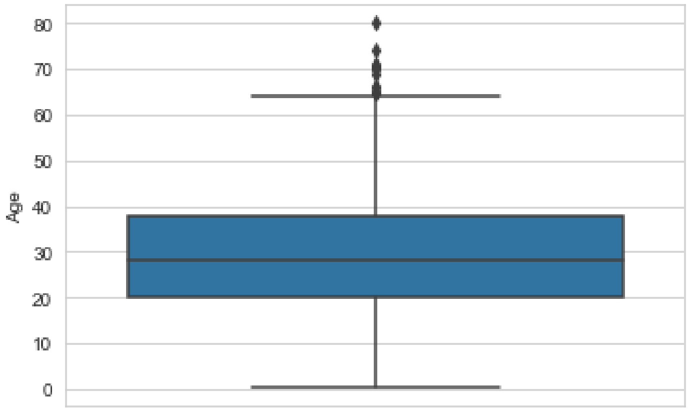
	Survived	Pclass	Name	Sex	Age	Siblings/Spouses Aboard	Parents/Children Aboard	Fare
0	0	3	Mr. Owen Harris Braund	male	22.0	1	0	7.2500
1	1	1	Mrs. John Bradley (Florence Briggs Thayer) Cum...	female	38.0	1	0	71.2833
2	1	3	Miss. Laina Heikkinen	female	26.0	0	0	7.9250
3	1	1	Mrs. Jacques Heath (Lily May Peel) Futrelle	female	35.0	1	0	53.1000
4	0	3	Mr. William Henry Allen	male	35.0	0	0	8.0500

In [122]:

```
sns.boxplot(train['Age'], orient = 'v')
```

Out[122]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d67a9ad308>
```



Bar Chart

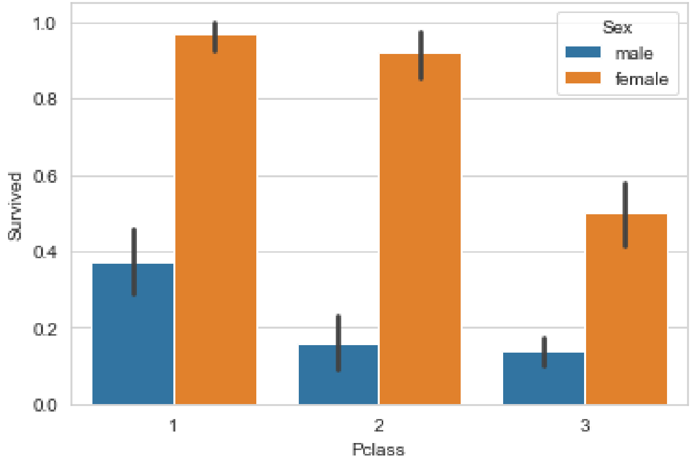
Bar charts are quite similar to line charts, i.e., they show a sequence of values. However, a bar is shown for each value, rather than points connected by lines. We can use the plt.bar function to draw a bar chart.

In [123]:

```
sns.barplot(x = 'Pclass', y='Survived', hue='Sex', data= train)
```

Out[123]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d67b9f1c48>
```



Conclusion

1) Female of passenger class 1st have more chance to survival as compared to male passenger of same class.

2) Within all female, passengers of 1 and 2 class have more chance to survival as compared to 3rd class.

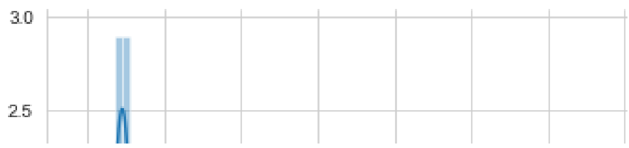
3) Within all male, passengers of 1st class have more survival rate as compared to other two classes.

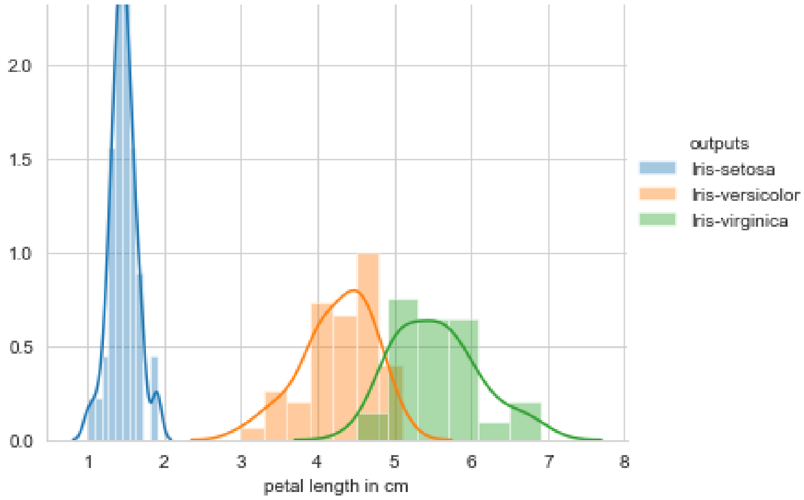
4) Female passenger of 1st and 2nd have almost same surivival rate.

5) Male of pclass 3 have very less chance of survival.

In [127]:

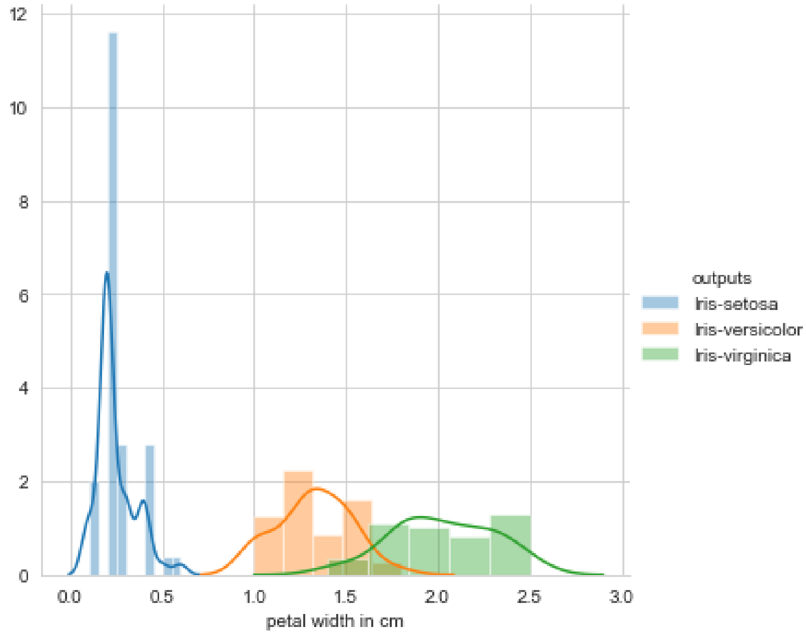
```
# Univariate Analysis

sns.FacetGrid(data, hue="outputs", size=5).map(sns.distplot, 'petal length in cm').add_le
gend()
plt.show()
```





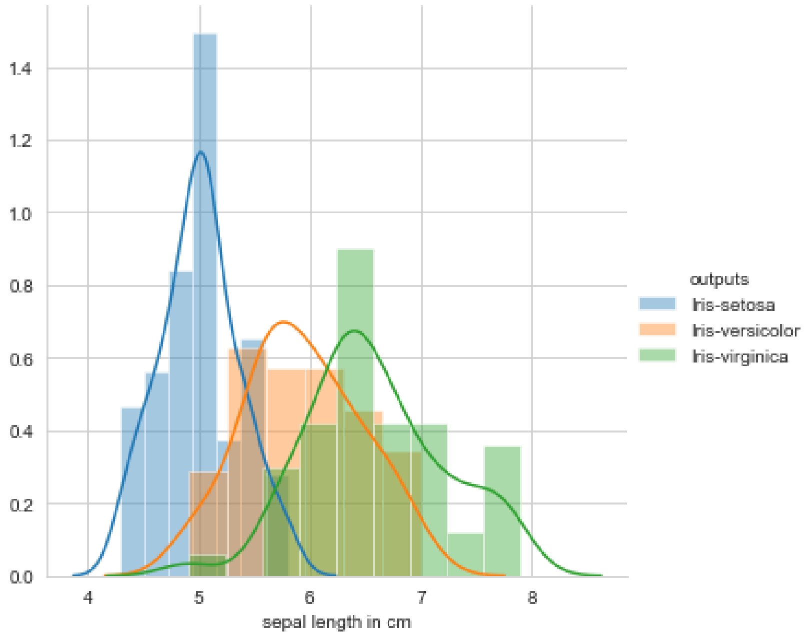
In [128]:

```
sns.FacetGrid(data, hue="outputs", size=5).map(sns.distplot, 'petal width in cm').add_leg
end()
plt.show()
```



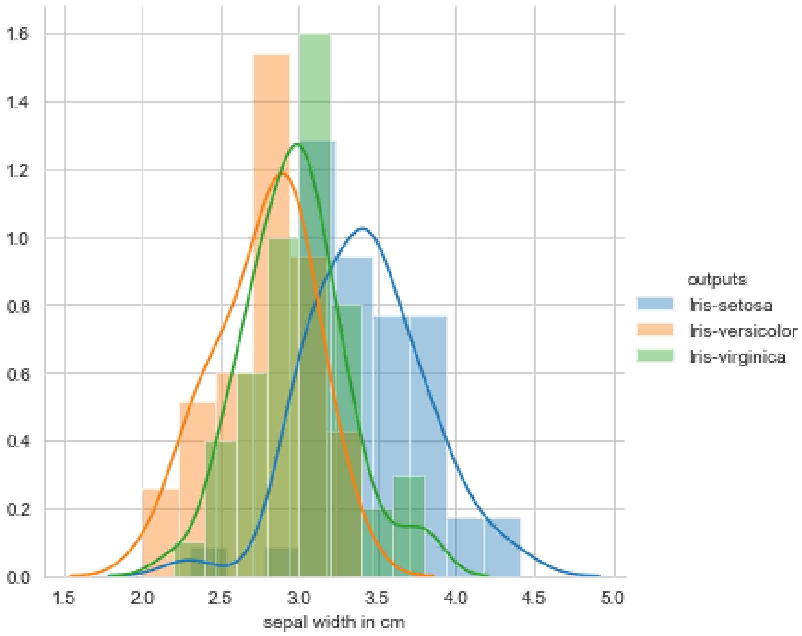
In [129]:

```
sns.FacetGrid(data, hue="outputs", size=5).map(sns.distplot, 'sepal length in cm').add_le
gend()
plt.show()
```



In [130]:

```
sns.FacetGrid(data, hue="outputs", size=5).map(sns.distplot, 'sepal width in cm').add_leg
end()
plt.show()
```



In [131]:

```
# Accessing the information from Setosa species

iris_setosa = data[data['outputs'] == 'Iris-setosa']
iris_setosa
```

Out[131]:

	sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	outputs
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa

	sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	outputs
20	5.4	3.4	1.7	0.2	Iris-setosa
21	5.1	3.7	1.5	0.4	Iris-setosa
22	4.6	3.6	1.0	0.2	Iris-setosa
23	5.1	3.3	1.7	0.5	Iris-setosa
24	4.8	3.4	1.9	0.2	Iris-setosa
25	5.0	3.0	1.6	0.2	Iris-setosa
26	5.0	3.4	1.6	0.4	Iris-setosa
27	5.2	3.5	1.5	0.2	Iris-setosa
28	5.2	3.4	1.4	0.2	Iris-setosa
29	4.7	3.2	1.6	0.2	Iris-setosa
30	4.8	3.1	1.6	0.2	Iris-setosa
31	5.4	3.4	1.5	0.4	Iris-setosa
32	5.2	4.1	1.5	0.1	Iris-setosa
33	5.5	4.2	1.4	0.2	Iris-setosa
34	4.9	3.1	1.5	0.2	Iris-setosa
35	5.0	3.2	1.2	0.2	Iris-setosa
36	5.5	3.5	1.3	0.2	Iris-setosa
37	4.9	3.6	1.4	0.1	Iris-setosa
38	4.4	3.0	1.3	0.2	Iris-setosa
39	5.1	3.4	1.5	0.2	Iris-setosa
40	5.0	3.5	1.3	0.3	Iris-setosa
41	4.5	2.3	1.3	0.3	Iris-setosa
42	4.4	3.2	1.3	0.2	Iris-setosa
43	5.0	3.5	1.6	0.6	Iris-setosa
44	5.1	3.8	1.9	0.4	Iris-setosa
45	4.8	3.0	1.4	0.3	Iris-setosa
46	5.1	3.8	1.6	0.2	Iris-setosa
47	4.6	3.2	1.4	0.2	Iris-setosa
48	5.3	3.7	1.5	0.2	Iris-setosa
49	5.0	3.3	1.4	0.2	Iris-setosa

In [132]:

```
# Versicolor

iris_versicolor = data[data['outputs'] == 'Iris-versicolor']
iris_versicolor
```

Out[132]:

	sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	outputs
50	7.0	3.2	4.7	1.4	Iris-versicolor
51	6.4	3.2	4.5	1.5	Iris-versicolor
52	6.9	3.1	4.9	1.5	Iris-versicolor
53	5.5	2.3	4.0	1.3	Iris-versicolor
54	6.5	2.8	4.6	1.5	Iris-versicolor
55	5.7	2.8	4.5	1.3	Iris-versicolor
56	6.3	3.3	4.7	1.6	Iris-versicolor
57	4.9	2.4	3.3	1.0	Iris-versicolor

	sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	outputs
58	6.6	2.9	4.6	1.3	Iris-versicolor
59	5.2	2.7	3.9	1.4	Iris-versicolor
60	5.0	2.0	3.5	1.0	Iris-versicolor
61	5.9	3.0	4.2	1.5	Iris-versicolor
62	6.0	2.2	4.0	1.0	Iris-versicolor
63	6.1	2.9	4.7	1.4	Iris-versicolor
64	5.6	2.9	3.6	1.3	Iris-versicolor
65	6.7	3.1	4.4	1.4	Iris-versicolor
66	5.6	3.0	4.5	1.5	Iris-versicolor
67	5.8	2.7	4.1	1.0	Iris-versicolor
68	6.2	2.2	4.5	1.5	Iris-versicolor
69	5.6	2.5	3.9	1.1	Iris-versicolor
70	5.9	3.2	4.8	1.8	Iris-versicolor
71	6.1	2.8	4.0	1.3	Iris-versicolor
72	6.3	2.5	4.9	1.5	Iris-versicolor
73	6.1	2.8	4.7	1.2	Iris-versicolor
74	6.4	2.9	4.3	1.3	Iris-versicolor
75	6.6	3.0	4.4	1.4	Iris-versicolor
76	6.8	2.8	4.8	1.4	Iris-versicolor
77	6.7	3.0	5.0	1.7	Iris-versicolor
78	6.0	2.9	4.5	1.5	Iris-versicolor
79	5.7	2.6	3.5	1.0	Iris-versicolor
80	5.5	2.4	3.8	1.1	Iris-versicolor
81	5.5	2.4	3.7	1.0	Iris-versicolor
82	5.8	2.7	3.9	1.2	Iris-versicolor
83	6.0	2.7	5.1	1.6	Iris-versicolor
84	5.4	3.0	4.5	1.5	Iris-versicolor
85	6.0	3.4	4.5	1.6	Iris-versicolor
86	6.7	3.1	4.7	1.5	Iris-versicolor
87	6.3	2.3	4.4	1.3	Iris-versicolor
88	5.6	3.0	4.1	1.3	Iris-versicolor
89	5.5	2.5	4.0	1.3	Iris-versicolor
90	5.5	2.6	4.4	1.2	Iris-versicolor
91	6.1	3.0	4.6	1.4	Iris-versicolor
92	5.8	2.6	4.0	1.2	Iris-versicolor
93	5.0	2.3	3.3	1.0	Iris-versicolor
94	5.6	2.7	4.2	1.3	Iris-versicolor
95	5.7	3.0	4.2	1.2	Iris-versicolor
96	5.7	2.9	4.2	1.3	Iris-versicolor
97	6.2	2.9	4.3	1.3	Iris-versicolor
98	5.1	2.5	3.0	1.1	Iris-versicolor
99	5.7	2.8	4.1	1.3	Iris-versicolor

In [133]:

```
# Virginica
```

```
iris_virginica = data[data['outputs'] == 'Iris-virginica']
iris_virginica
```

Out[133]:

	sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	outputs
100	6.3	3.3	6.0	2.5	Iris-virginica
101	5.8	2.7	5.1	1.9	Iris-virginica
102	7.1	3.0	5.9	2.1	Iris-virginica
103	6.3	2.9	5.6	1.8	Iris-virginica
104	6.5	3.0	5.8	2.2	Iris-virginica
105	7.6	3.0	6.6	2.1	Iris-virginica
106	4.9	2.5	4.5	1.7	Iris-virginica
107	7.3	2.9	6.3	1.8	Iris-virginica
108	6.7	2.5	5.8	1.8	Iris-virginica
109	7.2	3.6	6.1	2.5	Iris-virginica
110	6.5	3.2	5.1	2.0	Iris-virginica
111	6.4	2.7	5.3	1.9	Iris-virginica
112	6.8	3.0	5.5	2.1	Iris-virginica
113	5.7	2.5	5.0	2.0	Iris-virginica
114	5.8	2.8	5.1	2.4	Iris-virginica
115	6.4	3.2	5.3	2.3	Iris-virginica
116	6.5	3.0	5.5	1.8	Iris-virginica
117	7.7	3.8	6.7	2.2	Iris-virginica
118	7.7	2.6	6.9	2.3	Iris-virginica
119	6.0	2.2	5.0	1.5	Iris-virginica
120	6.9	3.2	5.7	2.3	Iris-virginica
121	5.6	2.8	4.9	2.0	Iris-virginica
122	7.7	2.8	6.7	2.0	Iris-virginica
123	6.3	2.7	4.9	1.8	Iris-virginica
124	6.7	3.3	5.7	2.1	Iris-virginica
125	7.2	3.2	6.0	1.8	Iris-virginica
126	6.2	2.8	4.8	1.8	Iris-virginica
127	6.1	3.0	4.9	1.8	Iris-virginica
128	6.4	2.8	5.6	2.1	Iris-virginica
129	7.2	3.0	5.8	1.6	Iris-virginica
130	7.4	2.8	6.1	1.9	Iris-virginica
131	7.9	3.8	6.4	2.0	Iris-virginica
132	6.4	2.8	5.6	2.2	Iris-virginica
133	6.3	2.8	5.1	1.5	Iris-virginica
134	6.1	2.6	5.6	1.4	Iris-virginica
135	7.7	3.0	6.1	2.3	Iris-virginica
136	6.3	3.4	5.6	2.4	Iris-virginica
137	6.4	3.1	5.5	1.8	Iris-virginica
138	6.0	3.0	4.8	1.8	Iris-virginica
139	6.9	3.1	5.4	2.1	Iris-virginica

	sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	outputs
140	6.7	3.1	5.6	2.4	Iris-virginica
141	6.9	3.1	5.1	2.3	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica
143	6.8	3.2	5.9	2.3	Iris-virginica
144	6.7	3.3	5.7	2.5	Iris-virginica
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

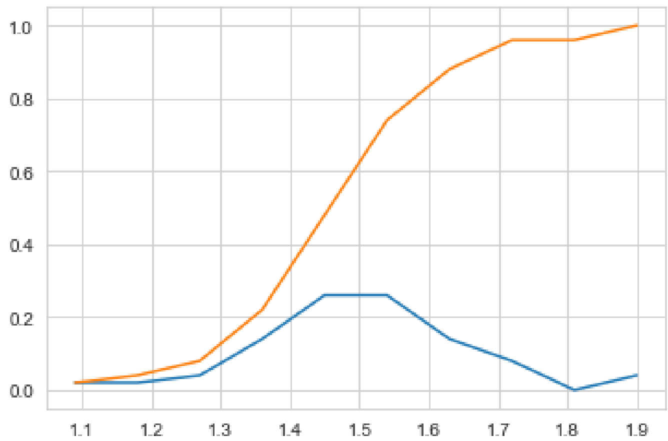
In [134]:

```
# Plot CDF and PDF of petal_length
# We can observe visually what percentage of setosa flower have a petal_length of less th
an 1.6?

counts, bin_edges = np.histogram(iris_setosa['petal length in cm'], bins = 10, density =
True)

pdf = counts / (sum(counts))
print(pdf)
print(bin_edges)

cdf = np.cumsum(pdf)

plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)
```

```
[0.02 0.02 0.04 0.14 0.26 0.26 0.14 0.08 0.   0.04]
[1.   1.09 1.18 1.27 1.36 1.45 1.54 1.63 1.72 1.81 1.9 ]
```

Out[134]:

```
[<matplotlib.lines.Line2D at 0x1d67bf1fcc8>]
```



In [136]:

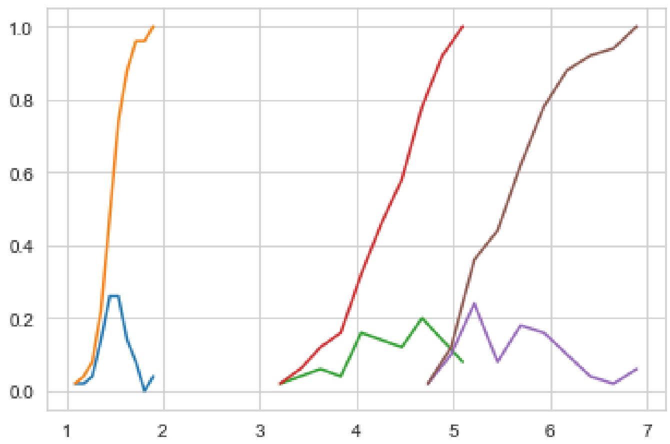
```
# Now Plots of CDF of petal_length for various types of flowers.

# For Setosa

counts, bin_edges = np.histogram(iris_setosa['petal length in cm'], bins = 10, density =
True)

pdf = counts / (sum(counts))
print(pdf)
print(bin_edges)

cdf = np.cumsum(pdf)
```

```
plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)

# For Versicolor

counts, bin_edges = np.histogram(iris_versicolor['petal length in cm'], bins = 10, densi
ty = True)

pdf = counts / (sum(counts))
print(pdf)
print(bin_edges)

cdf = np.cumsum(pdf)

plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)

# For Virginica

counts, bin_edges = np.histogram(iris_virginica['petal length in cm'], bins = 10, densit
y = True)

pdf = counts / (sum(counts))
print(pdf)
print(bin_edges)

#Compute CDF
cdf = np.cumsum(pdf)

plt.plot(bin_edges[1:], pdf)
plt.plot(bin_edges[1:], cdf)
```

```
[0.02 0.02 0.04 0.14 0.26 0.26 0.14 0.08 0.   0.04]
[1.   1.09 1.18 1.27 1.36 1.45 1.54 1.63 1.72 1.81 1.9 ]
[0.02 0.04 0.06 0.04 0.16 0.14 0.12 0.2  0.14 0.08]
[3.   3.21 3.42 3.63 3.84 4.05 4.26 4.47 4.68 4.89 5.1 ]
[0.02 0.1  0.24 0.08 0.18 0.16 0.1  0.04 0.02 0.06]
[4.5  4.74 4.98 5.22 5.46 5.7  5.94 6.18 6.42 6.66 6.9 ]
```

Out[136]:

```
[<matplotlib.lines.Line2D at 0x1d67bf8ecc8>]
```



In [137]:

```
np.mean(iris_setosa['petal length in cm'])
```

Out[137]:

```
1.4620000000000002
```

In [138]:

```
np.mean(np.append(iris_setosa['petal length in cm'], 50))
```

Out[138]:

```
2.4137254901960787
```

In [139]:

```
np.median(iris_setosa['petal length in cm'])
```

Out[139]:

1.5

In [140]:

```
np.median(np.append(iris_setosa['petal length in cm'], 50))
```

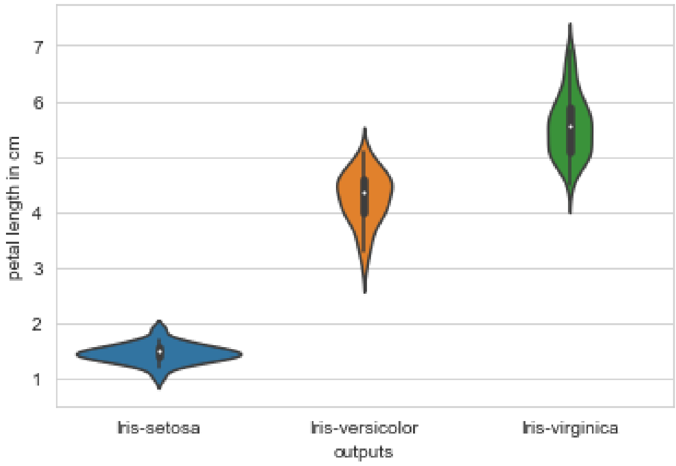
Out[140]:

1.5

Violin Plot

In [141]:

```
# Denser regions of the data are fatter, and sparser ones are thinner in a violin plot

sns.violinplot(x = 'outputs', y = 'petal length in cm', data = data, size = 7)
plt.show()
```



In [142]:

```
# Reading a wine dataset

wine = pd.read_csv("winequalityN.csv")
```

In [143]:

```
wine.head()
```

Out[143]:

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

In [144]:

```
# Finding the correlation matrix
```

```
corr = wine.corr()
corr
```

Out[144]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	q
fixed acidity	1.000000	0.220172	0.323736	-0.112319	0.298421	-0.283317	-0.329747	0.459204	-0.251814	0.300380	-0.095603	0.0
volatile acidity	0.220172	1.000000	-0.378061	-0.196702	0.377167	-0.353230	-0.414928	0.271193	0.260660	0.225476	-0.038248	0.2
citric acid	0.323736	-0.378061	1.000000	0.142486	0.039315	0.133437	0.195218	0.096320	-0.328689	0.057613	-0.010433	0.0
residual sugar	-0.112319	-0.196702	0.142486	1.000000	-0.128902	0.403439	0.495820	0.552498	-0.267050	-0.185745	-0.359706	0.0
chlorides	0.298421	0.377167	0.039315	-0.128902	1.000000	-0.195042	-0.279580	0.362594	0.044806	0.395332	-0.256861	0.2
free sulfur dioxide	-0.283317	-0.353230	0.133437	0.403439	-0.195042	1.000000	0.720934	0.025717	-0.145191	-0.188489	-0.179838	0.0
total sulfur dioxide	-0.329747	-0.414928	0.195218	0.495820	-0.279580	0.720934	1.000000	0.032395	-0.237687	-0.275381	-0.265740	0.0
density	0.459204	0.271193	0.096320	0.552498	0.362594	0.025717	0.032395	1.000000	0.011920	0.259454	-0.686745	0.3
pH	-0.251814	0.260660	-0.328689	-0.267050	0.044806	-0.145191	-0.237687	0.011920	1.000000	0.191248	0.121002	0.0
sulphates	0.300380	0.225476	0.057613	-0.185745	0.395332	-0.188489	-0.275381	0.259454	0.191248	1.000000	-0.003261	0.0
alcohol	-0.095603	-0.038248	-0.010433	-0.359706	-0.256861	-0.179838	-0.265740	-0.686745	0.121002	-0.003261	1.000000	0.4
quality	-0.077031	-0.265953	0.085706	-0.036825	-0.200886	0.055463	-0.041385	-0.305858	0.019366	0.038729	0.444319	1.0

Heatmap

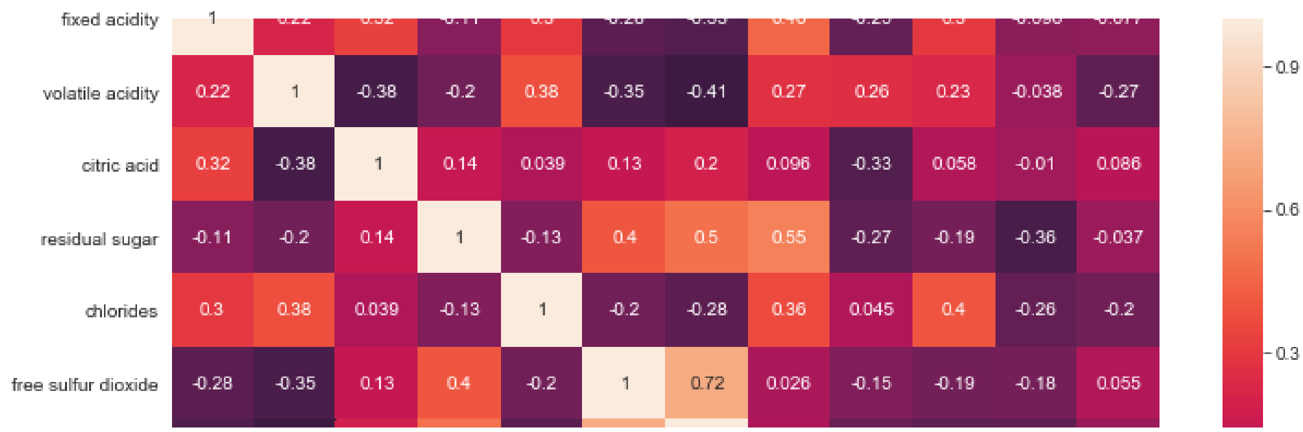
A heatmap is used to visualize 2-dimensional data like a matrix or a table using colors. The best way to understand it is by looking at an example.

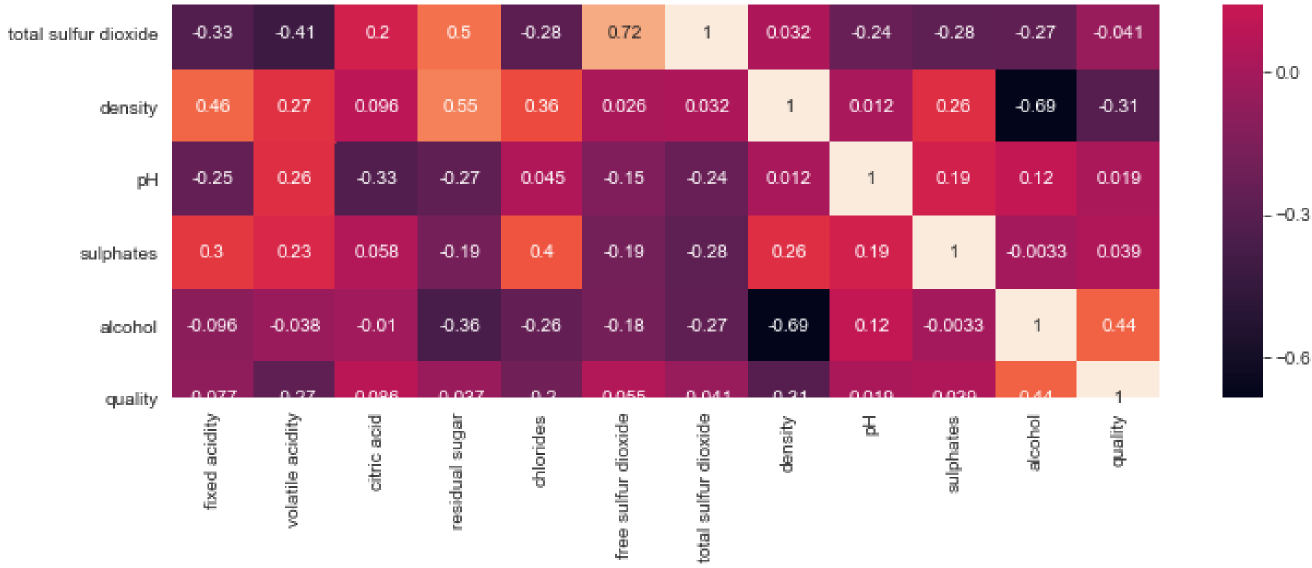
In [145]:

```
plt.figure(figsize = (12,8))
sns.heatmap(corr, annot=True)
```

Out[145]:

<matplotlib.axes._subplots.AxesSubplot at 0x1d67c0c9748>





total sulfur dioxide
density
pH
sulphates
alcohol
quality
fixed acidity
volatile acidity
citric acid
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
quality
0.0
−0.3
−0.6

In [ ]: