Iris Flower Classification

The categorization of iris flowers using machine learning is increasingly common. Each of the three flower classes in the iris dataset—**Versicolor, Setosa, and Virginica**—has four features: **sepal length, sepal width, petal length, and petal width**. The categorization of iris flowers aims to anticipate blooms based on their distinctive characteristics.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from warnings import filterwarnings
filterwarnings(action='ignore')
```

```
iris=pd.read_csv("/content/IRIS.csv")
```

Saving...

	sepal_length	sepal_width	petal_length	petal_width	species
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

```
iris.shape
```

```
(150, 5)
```

```
iris.describe()
```

	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

```
iris.isna().sum()
```

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

Saving...

```
iris.describe()
```

	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

```
iris.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
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

```
iris.head(150)
```

	sepal_length	sepal_width	petal_length	petal_width	species
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
...	...	...	...	...	...
			5.2	2.3	Iris-virginica
			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

Saving...

```
iris.tail(100)
```

	sepal_length	sepal_width	petal_length	petal_width	species
50	7.0	3.2	4.7	1.4	Iris-versicolor
51	6.4	3.2	4.5	1.5	Iris-versicolor

```
n = len(iris[iris['species'] == 'Iris-versicolor'])
```

```
n
```

```
50
```

145	6.7	3.0	5.2	2.3	Iris-virginica

```
n1 = len(iris[iris['species'] == 'Iris-virginica'])
```

```
n1
```

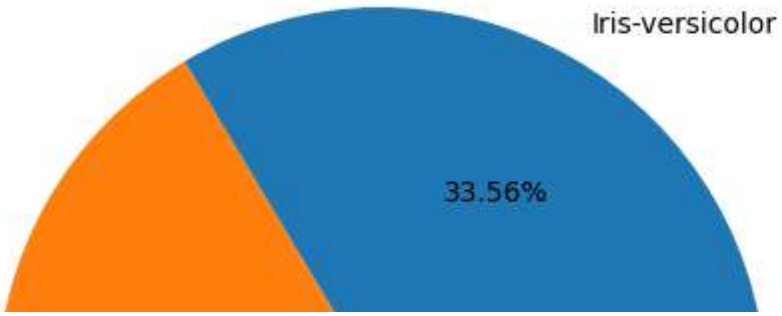
```
50
```

```
n2 = len(iris[iris['species'] == 'Iris-setosa'])
```

Saving...

```
50
```

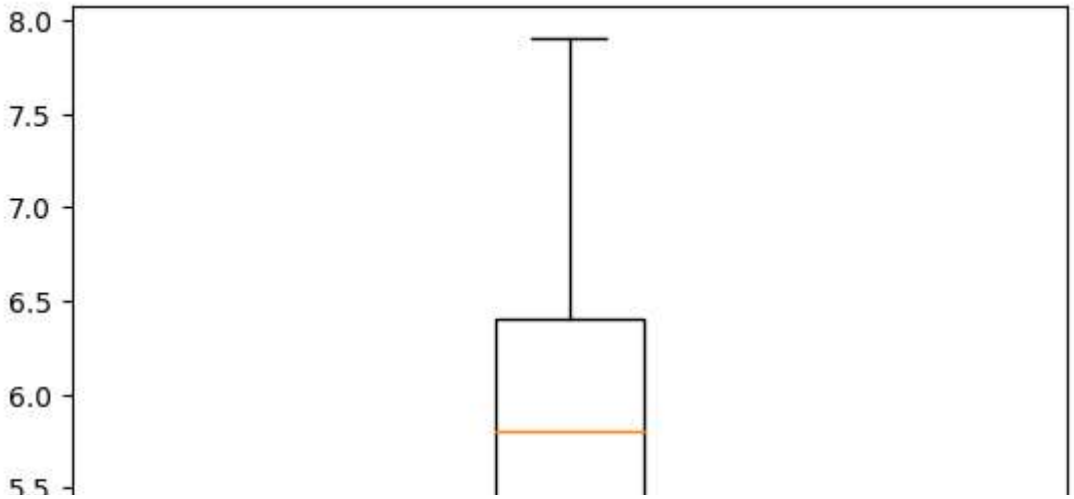
```
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.axis('equal')
l = ['Iris-versicolor', 'Iris-Setosa', 'Iris-Virginica']
s = [50,49,50]
ax.pie(s, labels = l, autopct = '%1.2f%%')
plt.show()
```



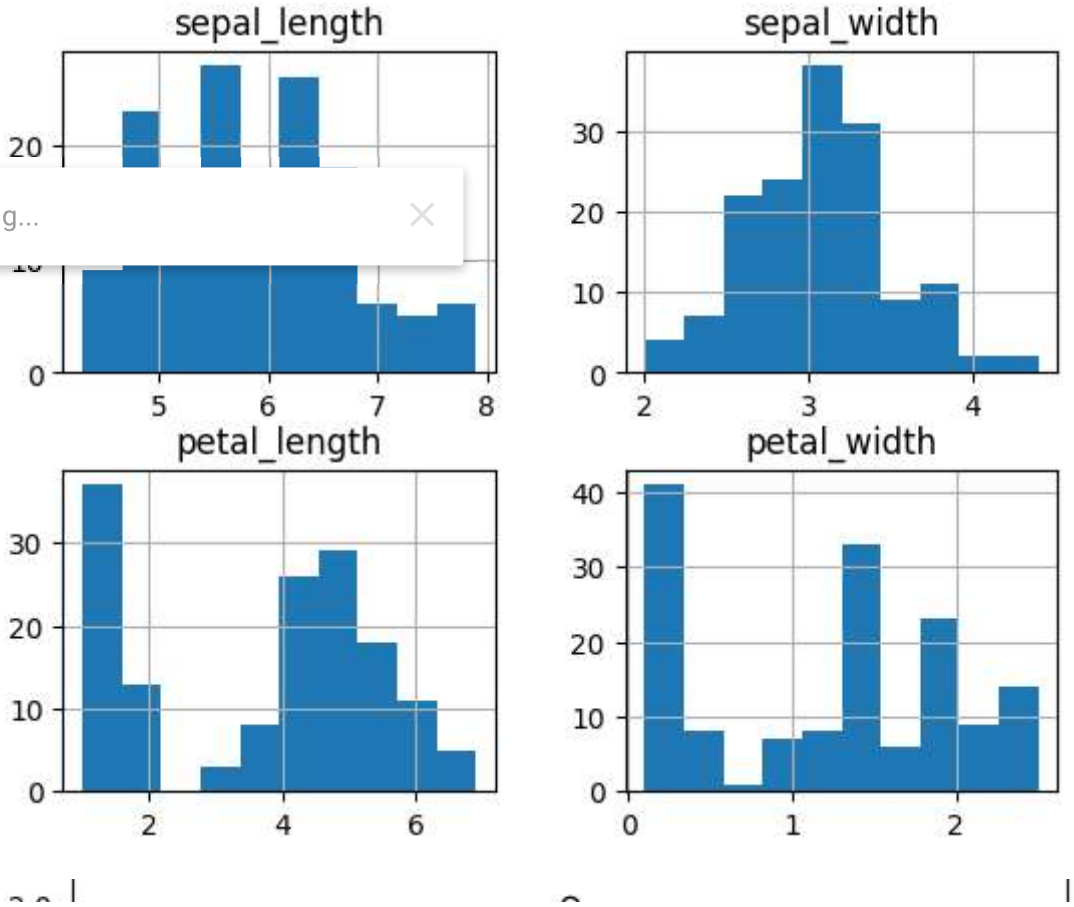

```
import matplotlib.pyplot as plt
```


```
plt.figure(1)
plt.boxplot([iris['sepal_length']])
plt.figure(2)
plt.boxplot([iris['sepal_width']])
plt.show()
```

Saving...



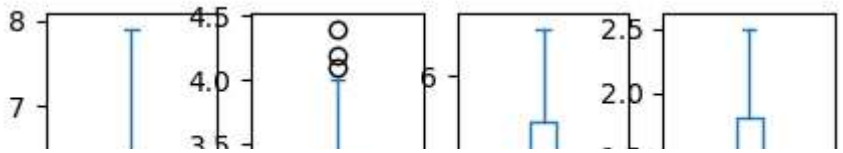

```
iris.hist()
plt.show()
```



Saving...

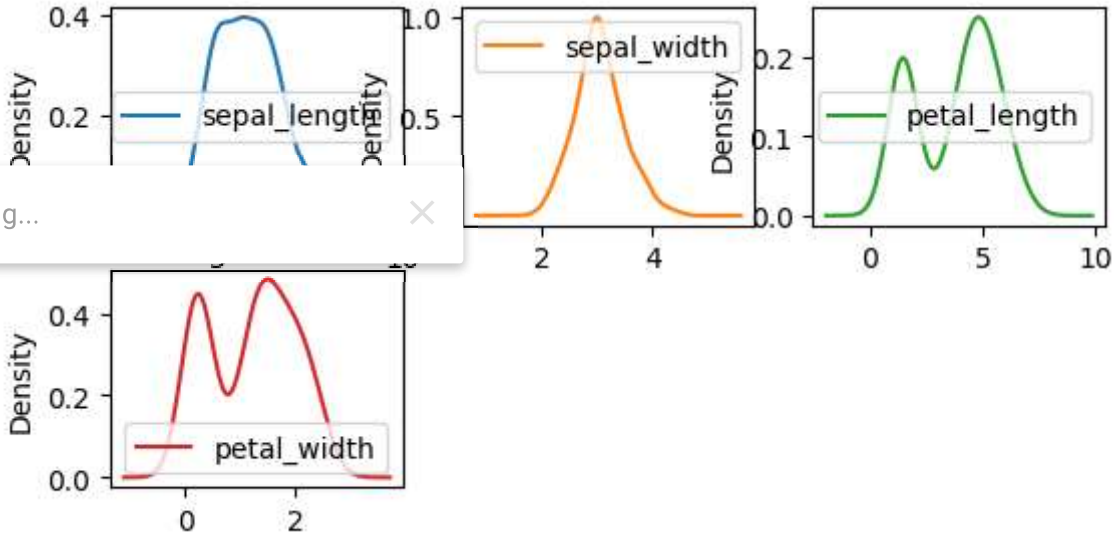
```
iris.plot(kind ='box',subplots = True, layout =(2,5),sharex = False)
```

```
sepal_length       Axes(0.125,0.53;0.133621x0.35)
sepal_width     Axes(0.285345,0.53;0.133621x0.35)
petal_length     Axes(0.44569,0.53;0.133621x0.35)
petal_width     Axes(0.606034,0.53;0.133621x0.35)
dtype: object
```



```
iris.plot(kind ='density',subplots = True, layout =(3,3),sharex = False)
```

```
array([[<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>]], dtype=object)
```



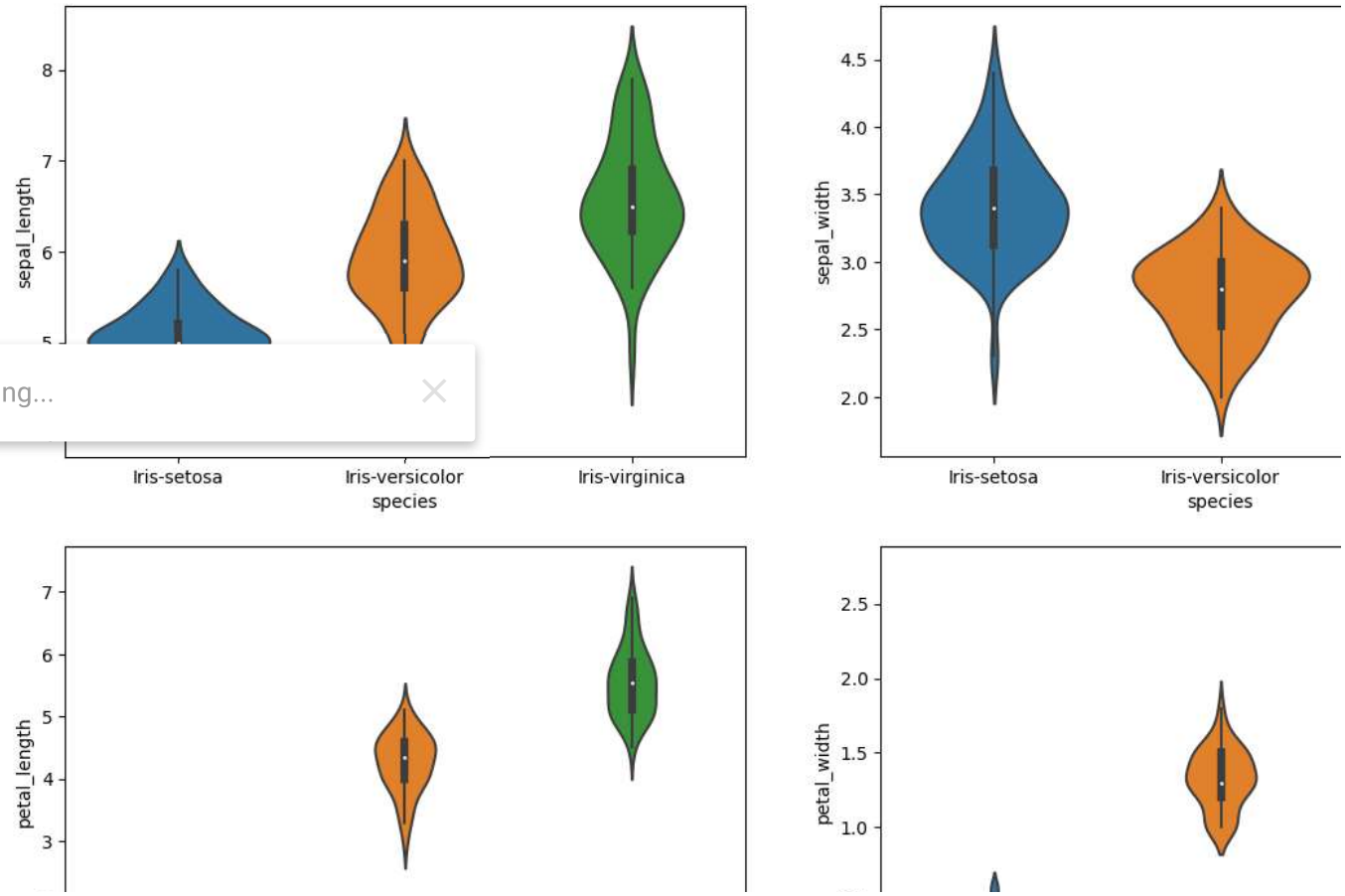
Saving...

```
iris.plot(kind ='box',subplots = True, layout =(2,5),sharex = False)
```

```
sepal_length       Axes(0.125,0.53;0.133621x0.35)
```

```
plt.figure(figsize=(15,10))
plt.subplot(2,2,1)
sns.violinplot(x='species',y='sepal_length',data=iris)
plt.subplot(2,2,2)
sns.violinplot(x='species',y='sepal_width',data=iris)
plt.subplot(2,2,3)
sns.violinplot(x='species',y='petal_length',data=iris)
plt.subplot(2,2,4)
sns.violinplot(x='species',y='petal_width',data=iris)
```

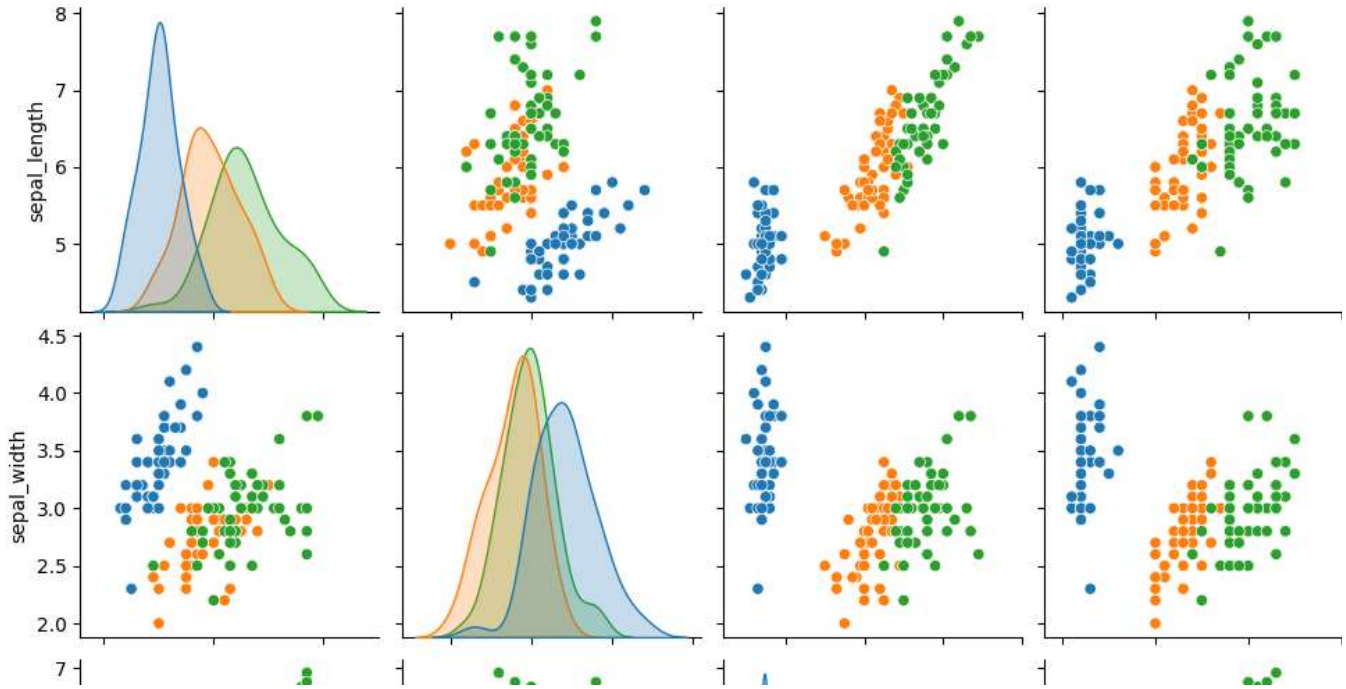
```
<Axes: xlabel='species', ylabel='petal_width'>
```



Saving...

```
sns.pairplot(iris,hue='species')
```

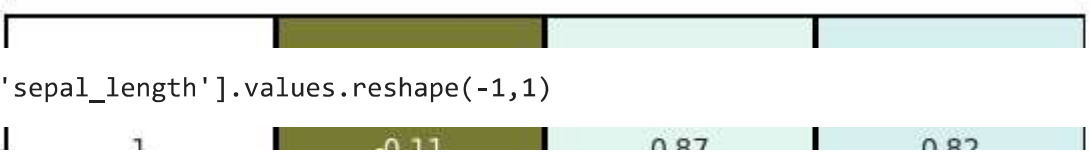
```
<seaborn.axisgrid.PairGrid at 0x7fe0a71eb9d0>
```




```
fig=plt.gcf()
fig.set_size_inches(10,7)
```

```
ue,cmap='cubehelix',linewidths=1,linecolor='k',
```

Saving...




```
X = iris['sepal_length'].values.reshape(-1,1)
```

```
X
```

Saving...

```
              [6.7],
              [6.9],
              [5.8],
              [6.8],
              [6.7],
              [6.7],
              [6.3],
              [6.5],
              [6.2],
              [5.9]])
```

```
Y = iris['sepal_length'].values.reshape(-1,1)
```

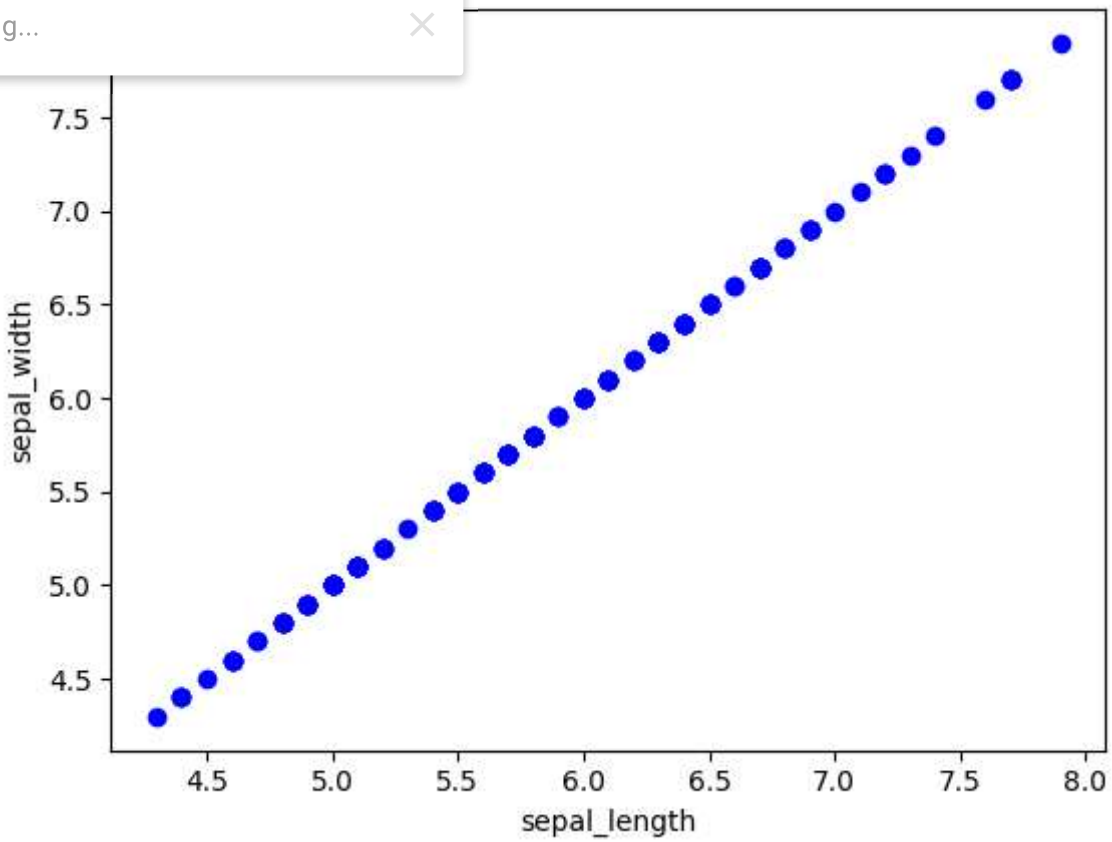
```
Y
```

Saving...

```
[6.4],
[6.3],
[6.1],
[7.7],
[6.3],
[6.4],
[6. ],
[6.9],
[6.7],
[6.9],
[5.8],
[6.8],
[6.7],
[6.7],
[6.3],
[6.5],
[6.2],
[5.9]])
```

```
plt.xlabel("sepal_length")
plt.ylabel("sepal_width")
plt.scatter(X,Y,color='b')
plt.show()
```

Saving...




```
corr_mat = iris.corr()
```

```
corr_mat
```

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.109369	0.871754	0.817954
sepal_width	-0.109369	1.000000	-0.420516	-0.356544
petal_length	0.871754	-0.420516	1.000000	0.962757
petal_width	0.817954	-0.356544	0.962757	1.000000

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier


train, test = train_test_split(iris, test_size = 0.25)


train.shape
```

Saving...

```
test.shape
```

```
(38, 5)
```

```
train_X = train[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
train_y = train.species
test_X = test[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
test_y = test.species


train_X.head()
```

	sepal_length	sepal_width	petal_length	petal_width
121	5.6	2.8	4.9	2.0
13	4.3	3.0	1.1	0.1
83	6.0	2.7	5.1	1.6
71	6.1	2.8	4.0	1.3
31	5.4	3.4	1.5	0.4

```
test_y.head()
```

```
125     Iris-virginica
147     Iris-virginica
48         Iris-setosa
135     Iris-virginica
94     Iris-versicolor
Name: species, dtype: object
```

```
model = LogisticRegression()
model.fit(train_X, train_y)
prediction = model.predict(test_X)
print('Accuracy:',metrics.accuracy_score(prediction,test_y))
```

```
Accuracy: 1.0
```

```
from sklearn.metrics import confusion_matrix,classification_report
confusion_mat = confusion_matrix(test_y,prediction)
print("Confusion matrix: \n",confusion_mat)
print(classification_report(test_y,prediction))
```

```
Confusion matrix:
 [[13  0  0]
```

Saving...

```
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        13
Iris-versicolor       1.00      1.00      1.00        13
 Iris-virginica       1.00      1.00      1.00        12

       accuracy                           1.00        38
      macro avg       1.00      1.00      1.00        38
   weighted avg       1.00      1.00      1.00        38
```

```
from sklearn.svm import SVC
model1 = SVC()
model1.fit(train_X,train_y)
pred_y = model1.predict(test_X)
from sklearn.metrics import accuracy_score
print("Acc=",accuracy_score(test_y,pred_y))
```

```
Acc= 1.0
```

```
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=5)
model2.fit(train_X,train_y)
y_pred2 = model2.predict(test_X)
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(test_y,y_pred2))
```

```
Accuracy Score: 1.0
```

```
from sklearn.naive_bayes import GaussianNB
model3 = GaussianNB()
model3.fit(train_X,train_y)
y_pred3 = model3.predict(test_X)
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(test_y,y_pred3))
```

```
Accuracy Score: 0.9736842105263158
```

```
from sklearn.tree import DecisionTreeClassifier
model4 = DecisionTreeClassifier(criterion='entropy',random_state=7)
model4.fit(train_X,train_y)
y_pred4 = model4.predict(test_X)
from sklearn.metrics import accuracy_score
print("Accuracy Score:",accuracy_score(test_y,y_pred4))
```

```
Accuracy Score: 1.0
```

Saving...

✓ 0s completed at 9:26 PM