In [1]:

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

In [2]:

```
iris =pd.read_csv(r'C:\Users\786\Downloads\IRIS.csv')
iris
```

Out[2]:

	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

In [3]:

```
iris.head()
```

Out[3]:

	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

In [4]:

```
iris.describe()
```

Out[4]:

	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

In [5]:

```
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 [6]:

```
iris['species'].value_counts()
```

Out[6]:

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

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

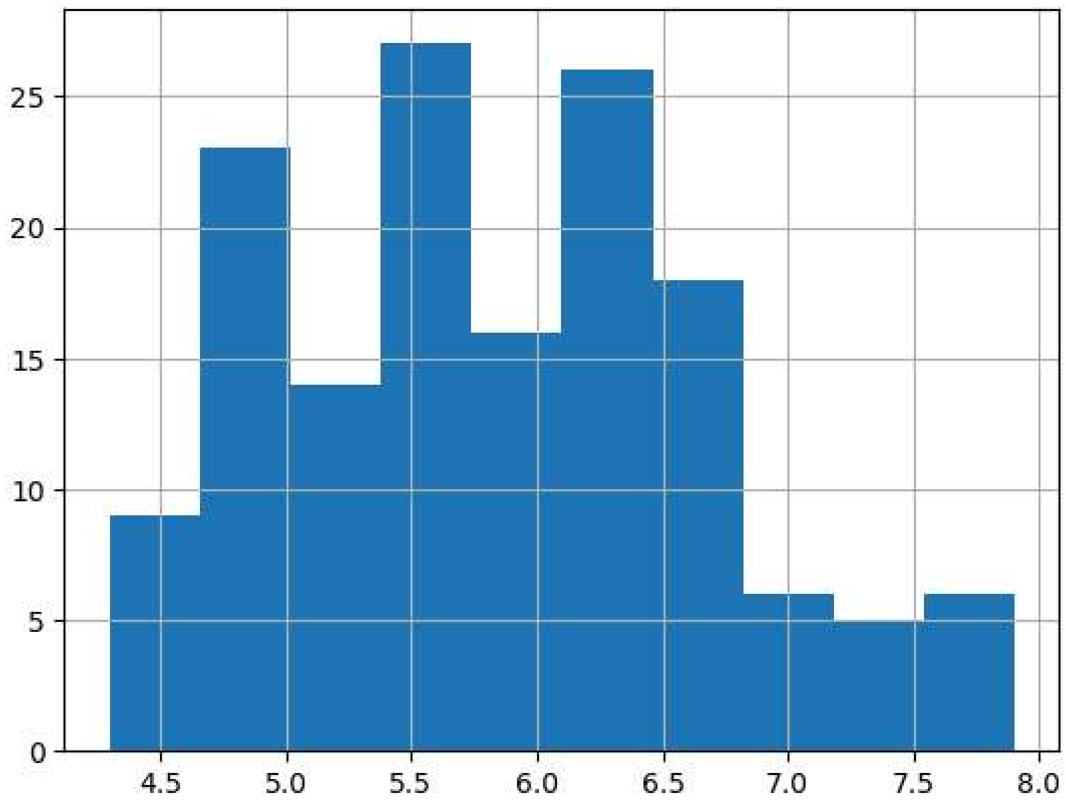
Out[7]:
```
5.0    10
5.1     9
6.3     9
5.7     8
6.7     8
5.8     7
5.5     7
6.4     7
4.9     6
5.4     6
6.1     6
6.0     6
5.6     6
4.8     5
6.5     5
6.2     4
7.7     4
6.9     4
4.6     4
5.2     4
5.9     3
4.4     3
7.2     3
6.8     3
6.6     2
4.7     2
7.6     1
7.4     1
7.3     1
7.0     1
7.1     1
5.3     1
4.3     1
4.5     1
7.9     1
Name: sepal_length, dtype: int64
```

In [8]:
```
iris.isnull().sum()
```

Out[8]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

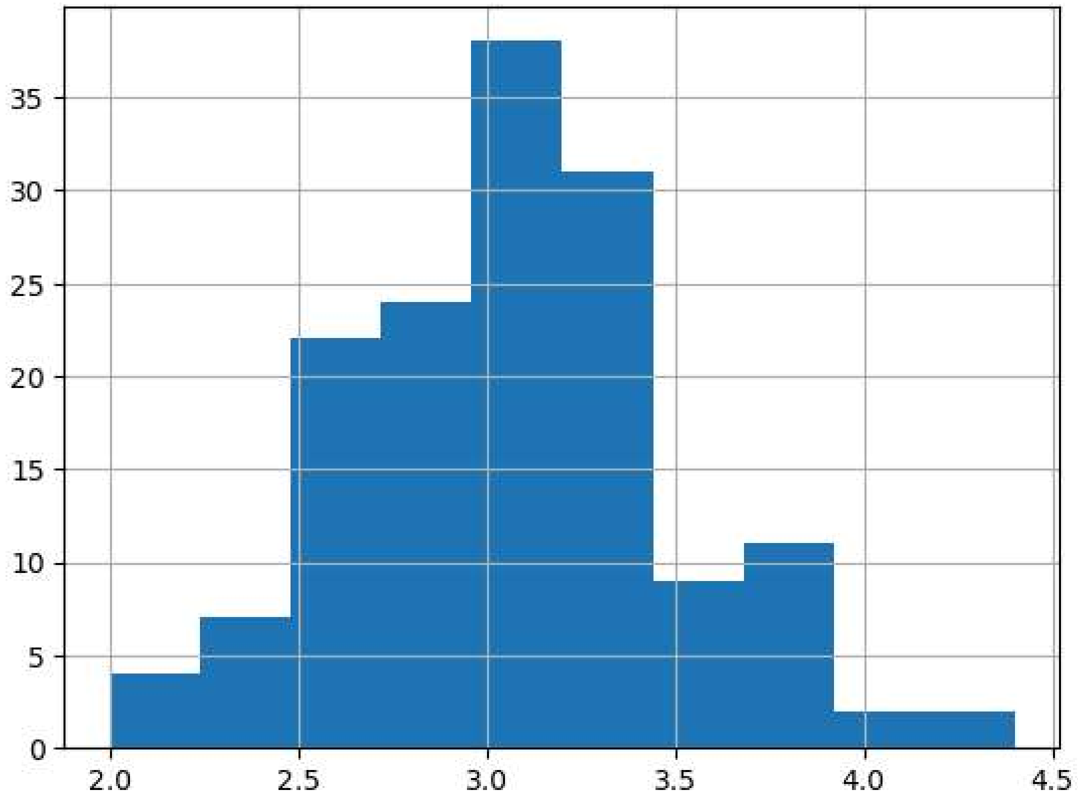
```
In [9]: iris['sepal_length'].hist()

Out[9]: <AxesSubplot:>
```



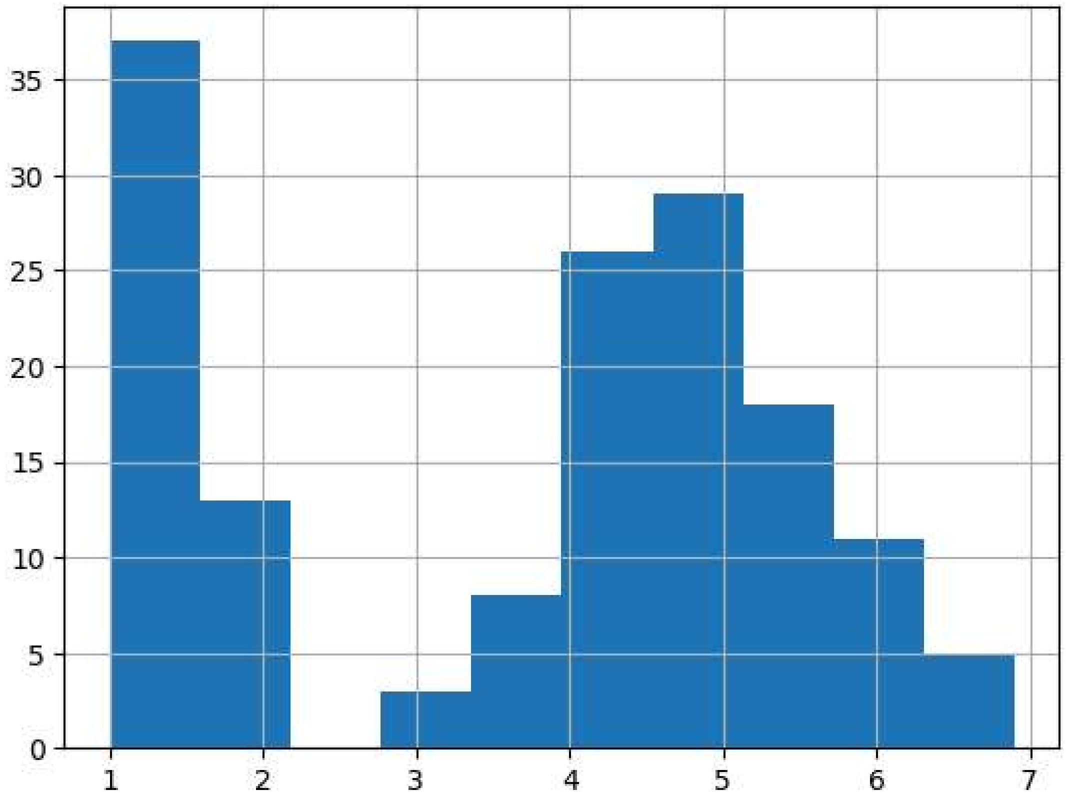
```
In [10]: iris['sepal_width'].hist()
```

```
Out[10]: <AxesSubplot:>
```



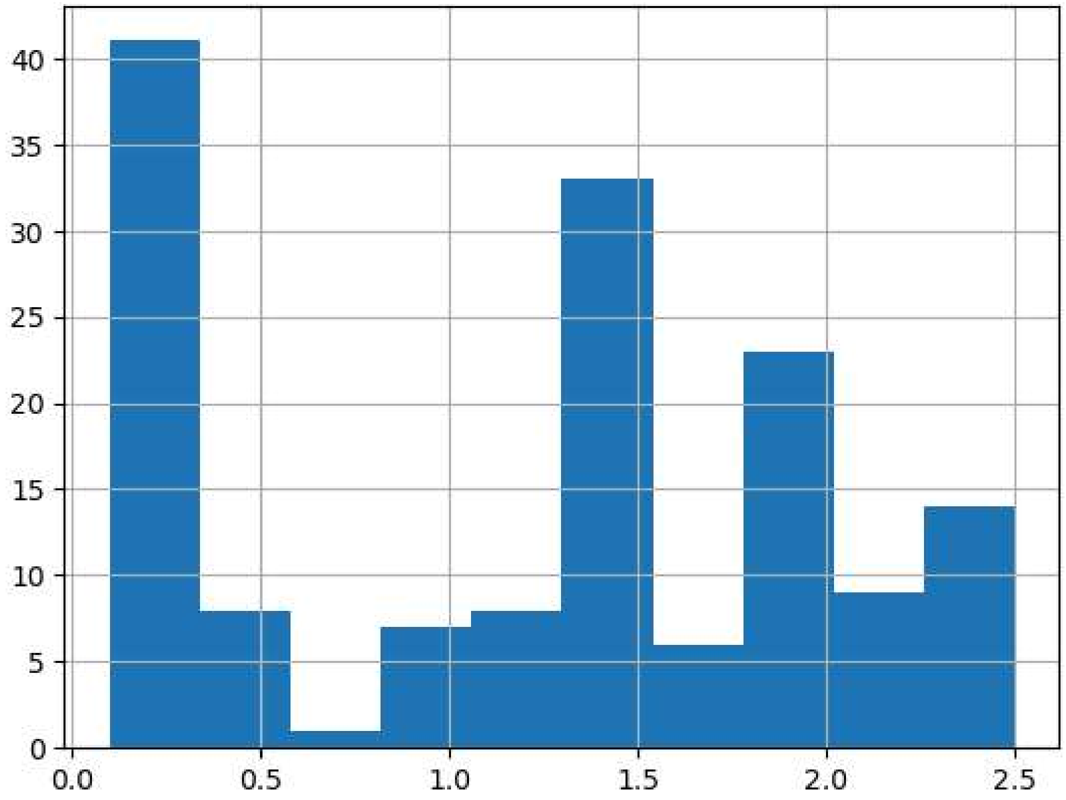
```
In [11]: iris['petal_length'].hist()
```

```
Out[11]: <AxesSubplot:>
```



In [12]:
```
iris['petal_width'].hist()
```

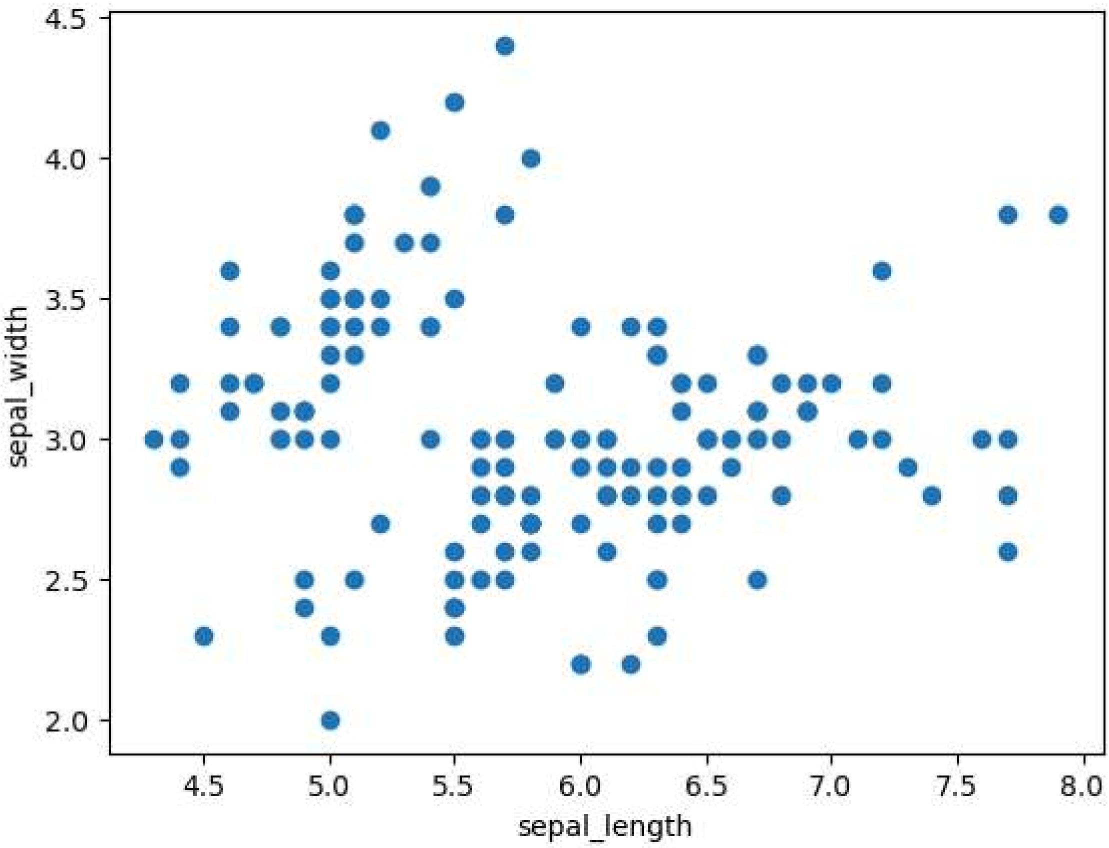
Out[12]:
```
<AxesSubplot:>
```



In [13]:
```
colors = ['red','blue','green']
species = ['Iris_virginica','Iris_versicolor','Iris_setosa']
```

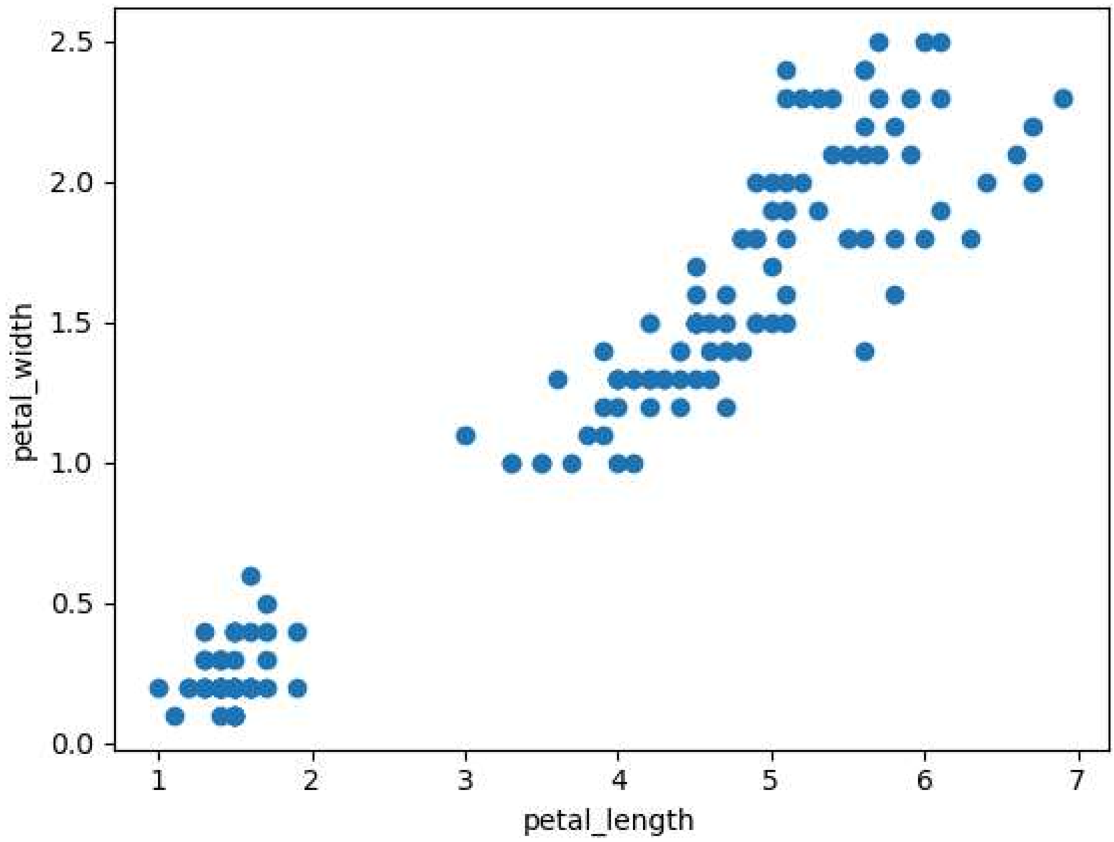
In [14]:

```
x=iris['sepal_length']
y=iris['sepal_width']
plt.xlabel('sepal_length')
plt.ylabel('sepal_width')
plt.scatter(x,y)
plt.show()
```



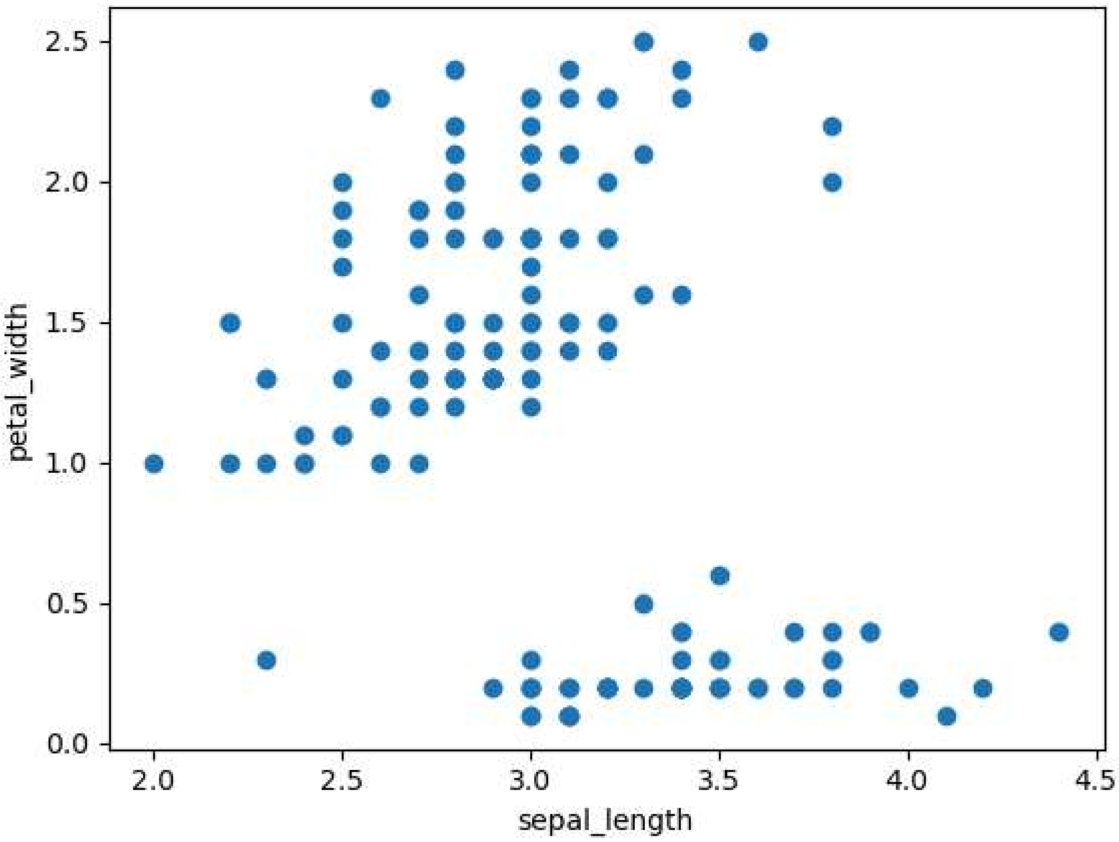
In [15]:

```
x=iris['petal_length']
y=iris['petal_width']
plt.xlabel('petal_length')
plt.ylabel('petal_width')
plt.scatter(x,y)
plt.show()
```



In [16]:

```
x=iris['sepal_width']
y=iris['petal_width']
plt.xlabel('sepal_length')
plt.ylabel('petal_width')
plt.scatter(x,y)
plt.show()
```



In [17]:

```
iris.corr()
```

Out[17]:

	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

In [18]:

```
pd.get_dummies(iris['species']).head()
```

Out[18]:

	Iris-setosa	Iris-versicolor	Iris-virginica
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0

In [19]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [20]:

```
iris['species'] = le.fit_transform(iris['species'])
iris.head()
```

Out[20]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [21]:

```
x = iris.drop(columns=['species'],axis=1)
y = iris['species']
```

In [22]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30)
```

In [23]:

```
print(x_train)
```

```
     sepal_length  sepal_width  petal_length  petal_width
5             5.4          3.9           1.7          0.4
37            4.9          3.1           1.5          0.1
43            5.0          3.5           1.6          0.6
111           6.4          2.7           5.3          1.9
6             4.6          3.4           1.4          0.3
..            ...          ...           ...          ...
132           6.4          2.8           5.6          2.2
12            4.8          3.0           1.4          0.1
56            6.3          3.3           4.7          1.6
31            5.4          3.4           1.5          0.4
123           6.3          2.7           4.9          1.8

[105 rows x 4 columns]
```

In [24]:
```
print(y_train)
```

```
5      0
37     0
43     0
111    2
6      0
      ..
132    2
12     0
56     1
31     0
123    2
Name: species, Length: 105, dtype: int32
```

In [25]:
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
```

In [26]:
```
logreg.fit(x_train, y_train)
```

Out[26]:
```
LogisticRegression()
```

In [27]:
```
y_pred = logreg.predict(x_test)
```

In [28]:
```
from sklearn.metrics import confusion_matrix
```

In [29]:
```
confusion_matrix(y_test,y_pred)
```

Out[29]:
```
array([[14,  0,  0],
       [ 0, 15,  1],
       [ 0,  1, 14]], dtype=int64)
```

In [30]:
```
from sklearn.metrics import accuracy_score
```

In [31]:
```
accuracy_score(y_test,y_pred)
```

Out[31]:
```
0.9555555555555556
```

In [32]:
```
from sklearn.metrics import classification_report
```

In [33]:
```
classification_report(y_test,y_pred)
```

Out[33]:
```
'              precision    recall  f1-score   support\n\n           0
1.00      1.00      1.00        14\n           1       0.94      0.94
0.94        16\n           2       0.93      0.93      0.93        15\n\n
accuracy                           0.96        45\n   macro avg       0.96
0.96      0.96        45\nweighted avg       0.96      0.96      0.96
45\n'
```

In [34]:
```
x = iris.drop(columns=['species'],axis=1)
y = iris['species']
```

In [35]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30)
```

In [36]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
```

In [37]:
```
knn.fit(x_train,y_train)
```

Out[37]:
```
KNeighborsClassifier()
```

In [38]:
```
y_pred=knn.predict(x_test)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\neighbors\_classificati
on.py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `
kurtosis`), the default behavior of `mode` typically preserves the axis it
acts along. In SciPy 1.11.0, this behavior will change: the default value
of `keepdims` will become False, the `axis` over which the statistic is ta
ken will be eliminated, and the value None will no longer be accepted. Set
`keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [39]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[39]:
```
0.9555555555555556
```

In [40]:
```
x = iris.drop(columns=['species'],axis=1)
y = iris['species']
```

In [41]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30)
```

In [42]:
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
```

In [43]:
```
clf.fit(x_train,y_train)
```

Out[43]:
```
DecisionTreeClassifier()
```

In [44]:
```
y_pred = clf.predict(x_test)
```

In [45]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[45]:
```
0.9777777777777777
```

In [46]:

```
from sklearn.metrics import classification_report
classification_report(y_test,y_pred)
```

Out[46]:

```
'              precision    recall  f1-score   support\n\n           0
1.00      1.00      1.00        16\n           1       0.95      1.00
0.97        18\n           2       1.00      0.91      0.95        11\n\n
accuracy                           0.98        45\n   macro avg       0.98
0.97      0.98        45\nweighted avg       0.98      0.98      0.98
45\n'
```

In [ ]:

```
```