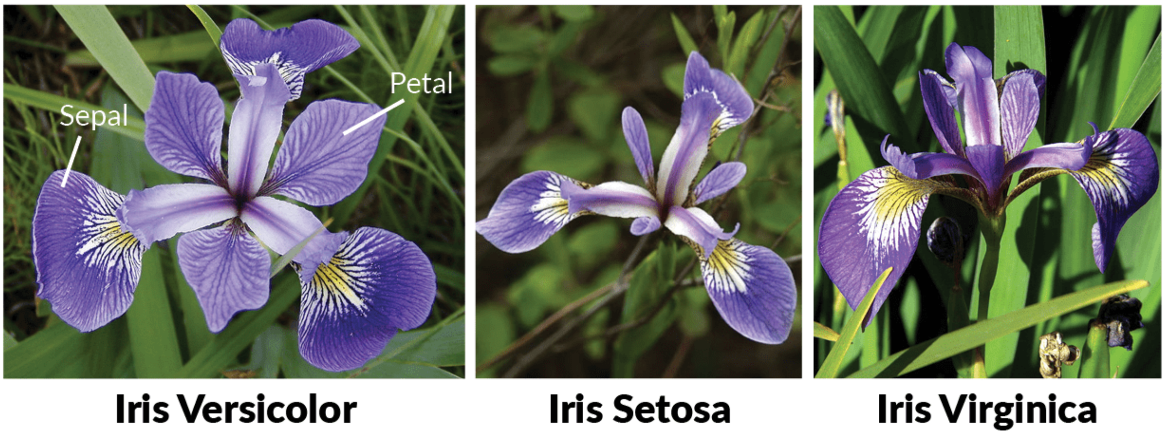
Iris_Flower_Classification



Importing Libraries

In [37]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

In [38]:
```
iris_data = pd.read_csv("iris.csv", names=['sepal_length', 'sepal_width', 'petal_length',
iris_data.head()
```

Out[38]:

	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 [7]:
```
iris_data.tail()
```

Out[7]:

	sepal_length	sepal_width	petal_length	petal_width	species
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

Statistical Data Analysis

In [8]:

```
iris_data.describe()
```

Out[8]:

	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 [9]:

```
#Length of Data
iris_data.shape
```

Out[9]:

```
(150, 5)
```

summary of a DataFrame

In [10]:

```
iris_data.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 [11]:

```
#Checking null value
iris_data.isnull().sum()
```

Out[11]:

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

In [12]:

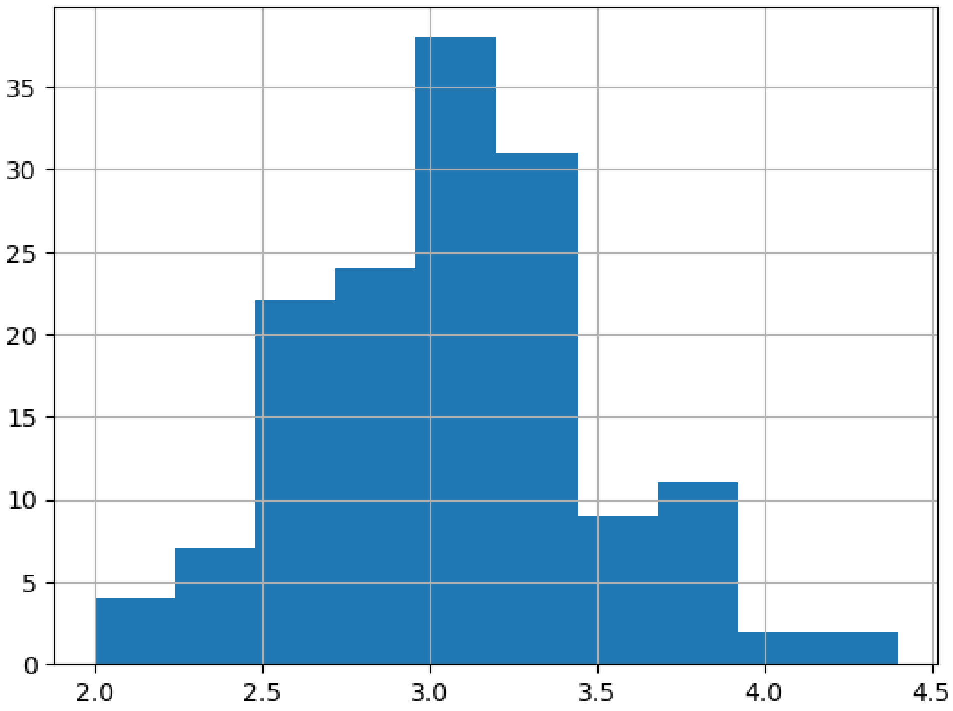
```
iris_data['species'].value_counts()
```

Out[12]:

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

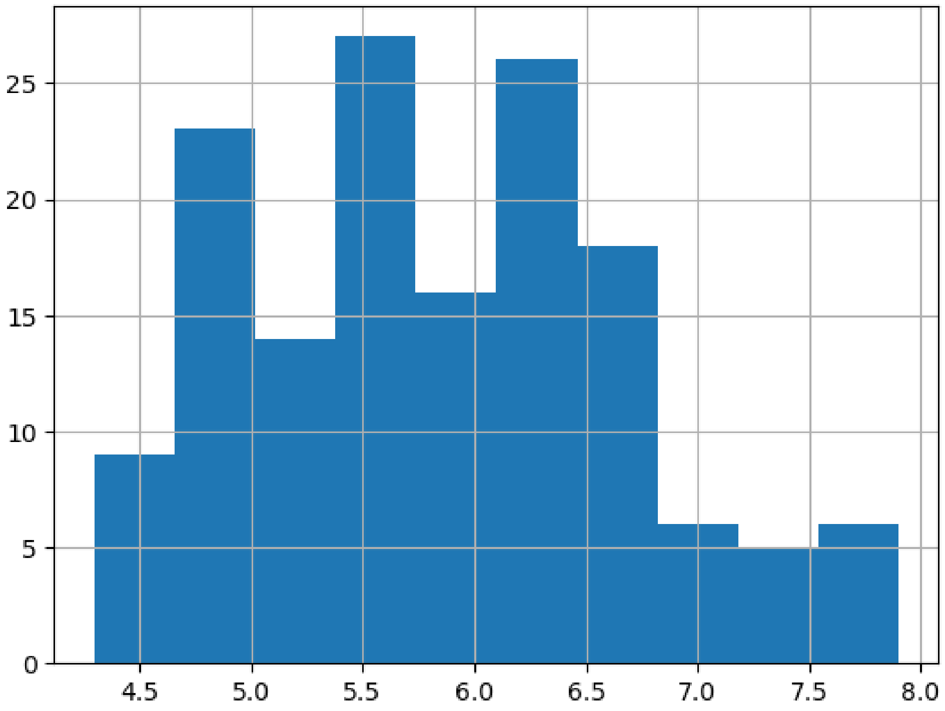
In [15]:

```
iris_data['sepal_width'].hist()
plt.show()
```



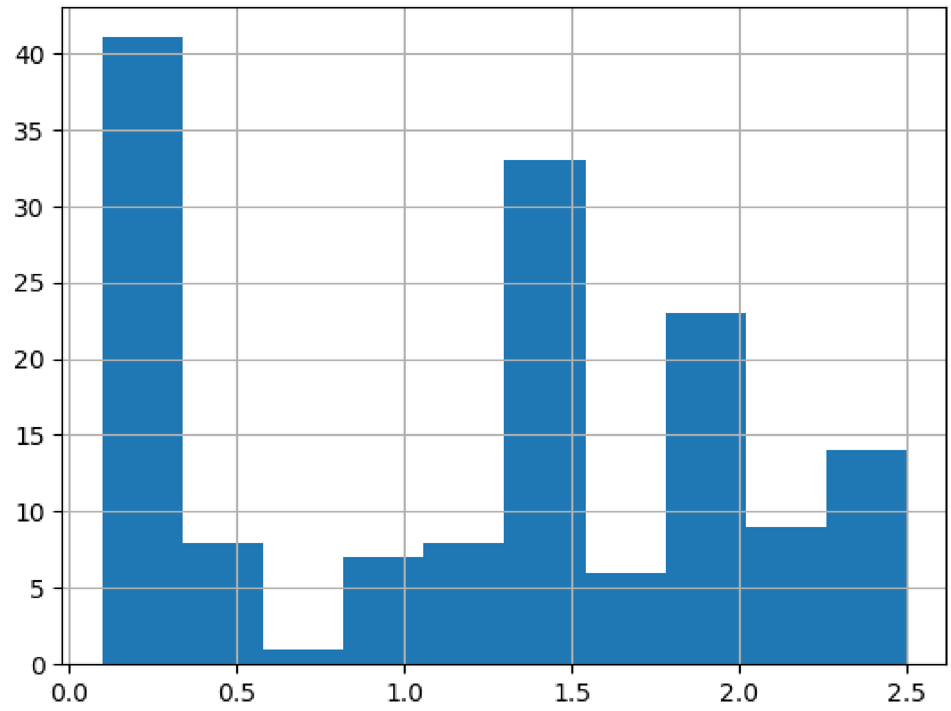
In [16]:

```
iris_data['sepal_length'].hist()
plt.show()
```



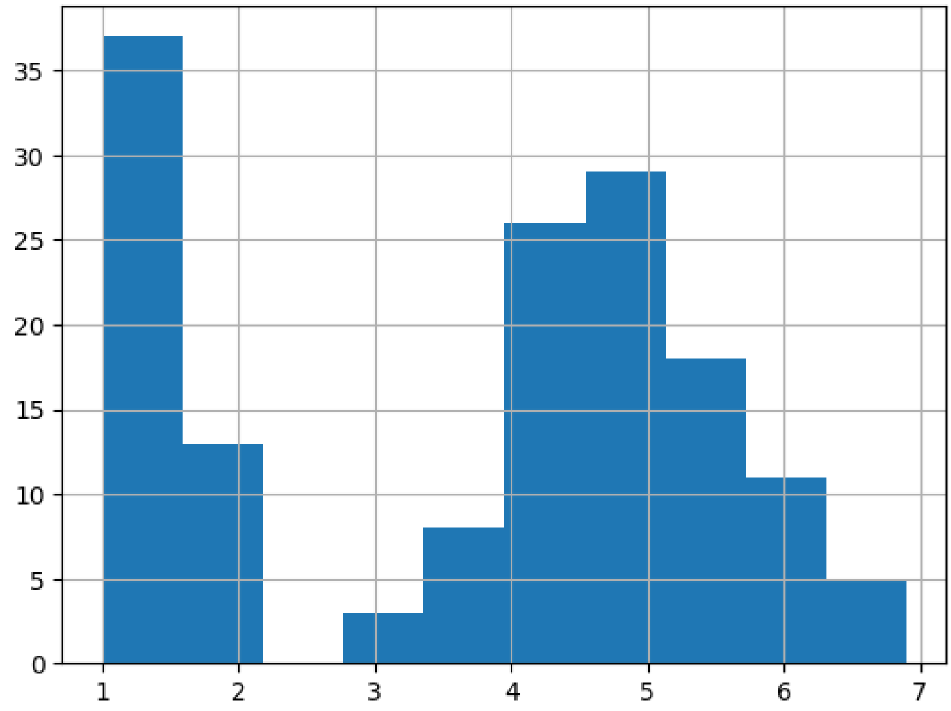
In [17]:

```
iris_data['petal_width'].hist()
plt.show()
```



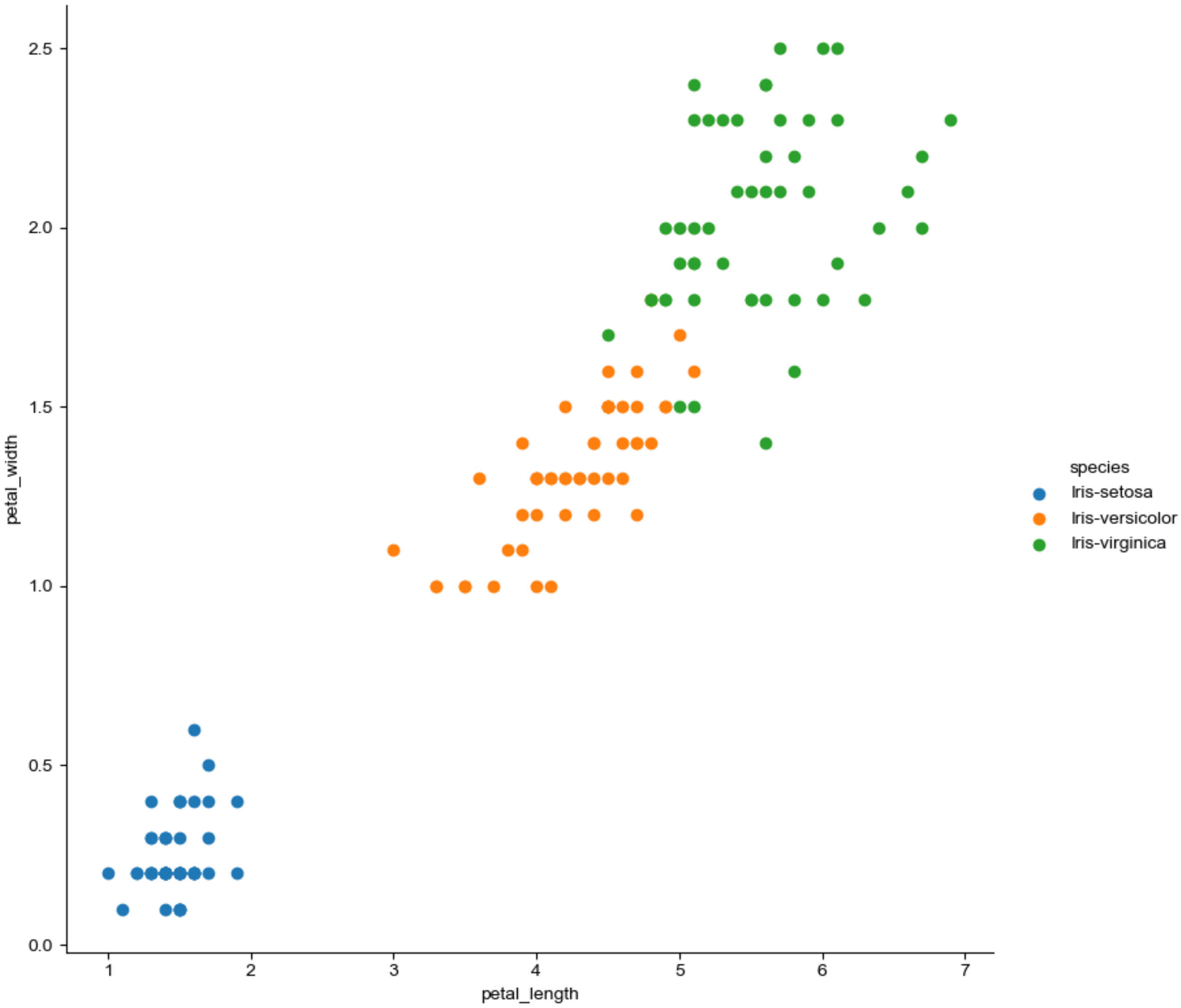
In [18]:

```
iris_data['petal_length'].hist()
plt.show()
```



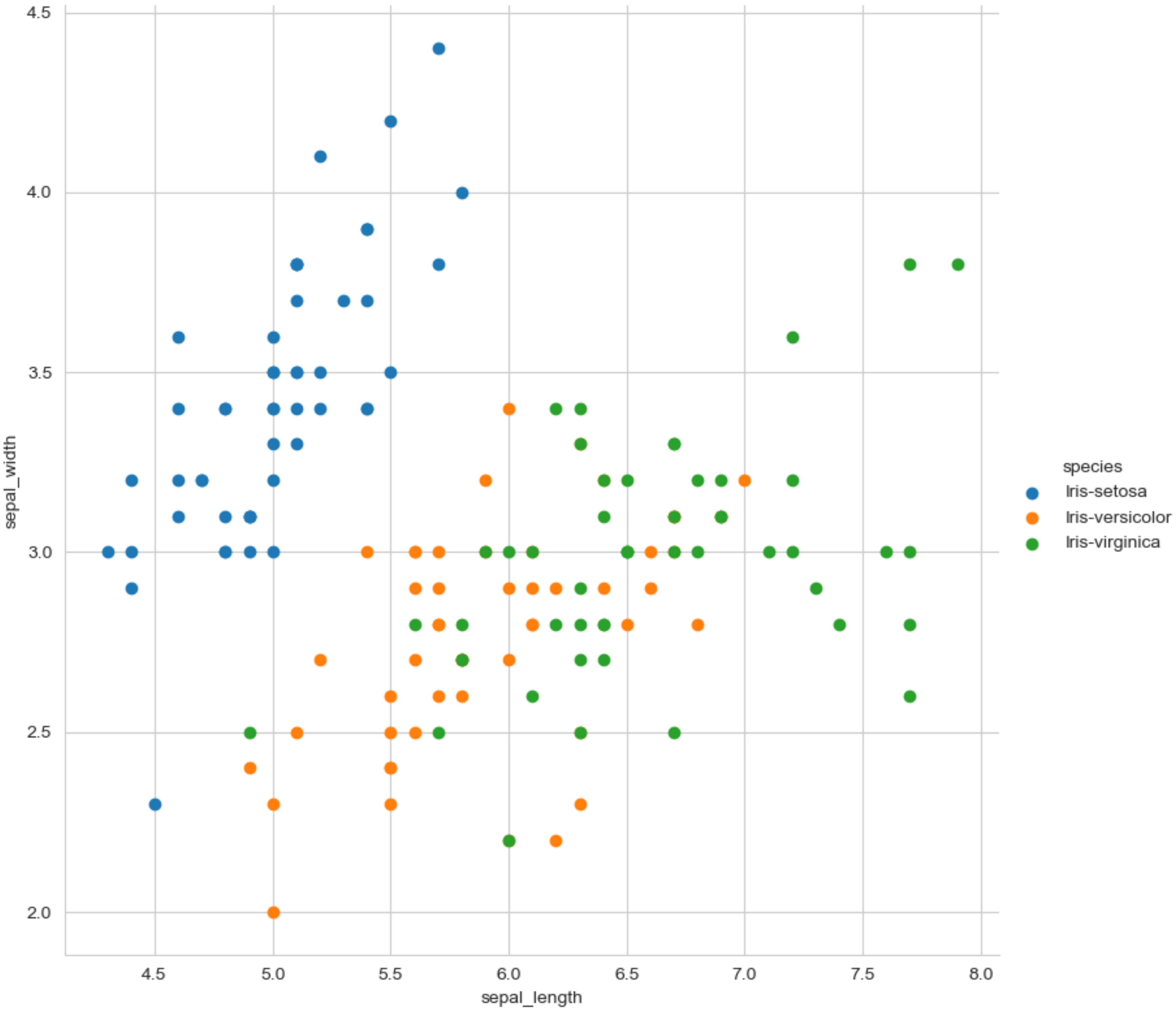
In [19]:

```
s = sns.FacetGrid(iris_data, height=8, hue="species")
s.map(plt.scatter, "petal_length", "petal_width")
s.add_legend()
sns.set_style("whitegrid")
plt.show()
```



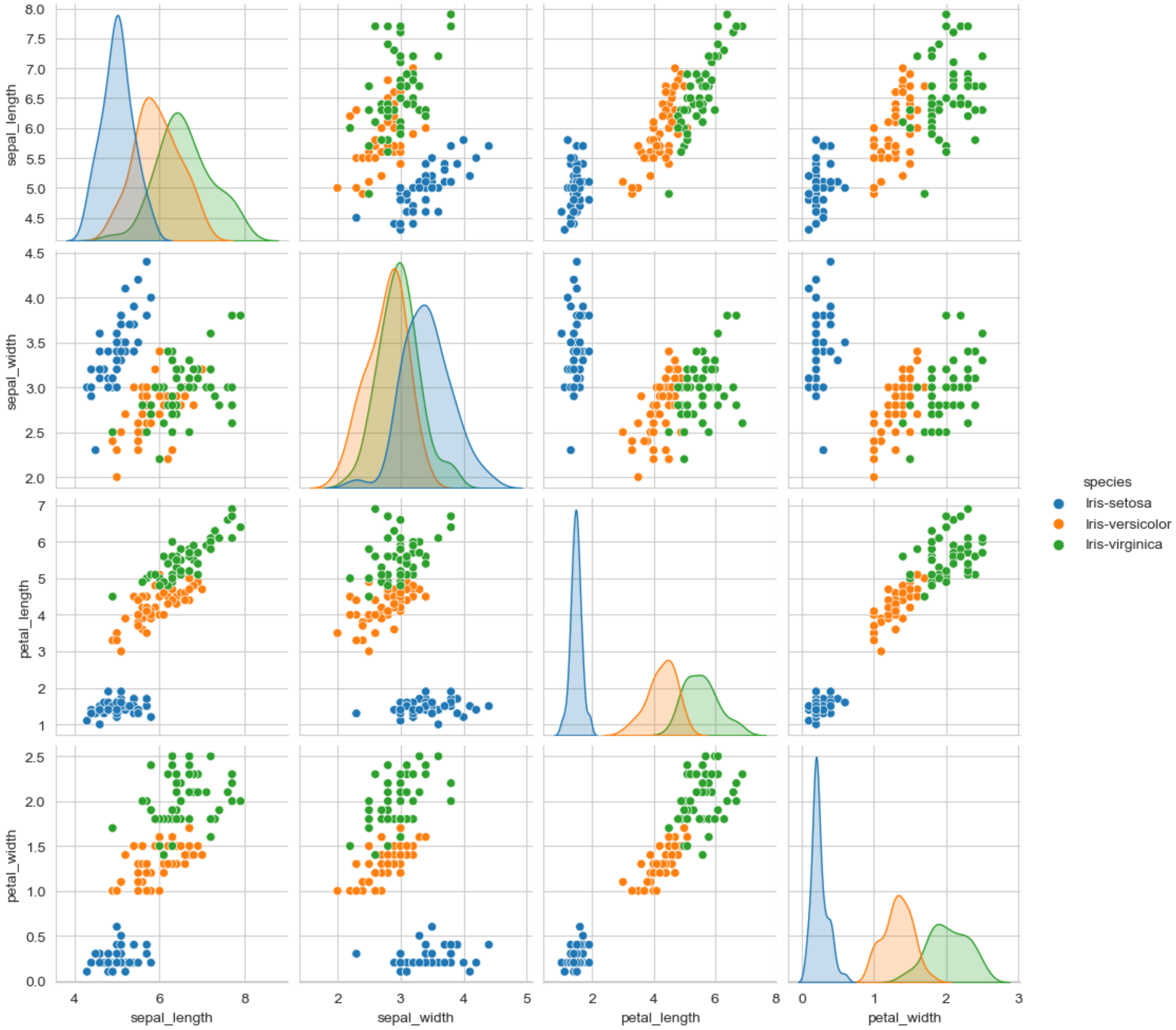
In [20]:

```
s = sns.FacetGrid(iris_data, height=8, hue="species")
s.map(plt.scatter, "sepal_length", "sepal_width")
s.add_legend()
sns.set_style("whitegrid")
plt.show()
```



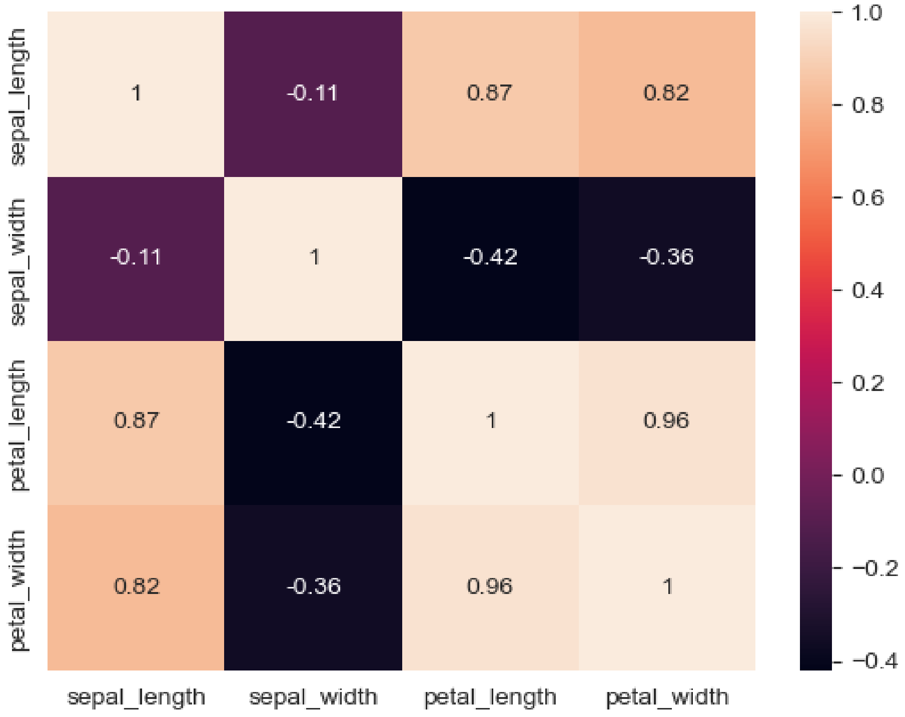
In [23]:

```
sns.pairplot(iris_data, height=2.5, hue="species")
plt.show()
```



In [22]:

```
#Checking Correlation use of Heatmap
sns.heatmap(iris_data.corr(), annot=True)
plt.show()
```



Split the data into training and testing

In [24]:
```
from sklearn.model_selection import train_test_split

X = iris_data[["sepal_length", "sepal_width", "petal_length", "petal_width"]]
y = iris_data["species"]
```

In [25]:
```
X
```

Out[25]:

	sepal_length	sepal_width	petal_length	petal_width
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 [26]:
```
y
```

Out[26]:
```
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: species, Length: 150, dtype: object
```

In [27]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=12
```

Logistic regression model

In [28]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

In [29]:
```
model.fit(X_train,y_train)
```

Out[29]:
```
▾ LogisticRegression
LogisticRegression()
```

In [30]:

```
#metrics to get performance
print('Accuracy',model.score(X_test,y_test)*100)
```

```
Accuracy 97.77777777777777
```

K-Nearest Neighbours model

In [31]:

```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier()
```

In [32]:

```
model.fit(X_train,y_train)
```

Out[32]:

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [33]:

```
#metrics to get performance
print('Accuracy',model.score(X_test,y_test)*100)
```

```
Accuracy 97.77777777777777
```

Decision tree model

In [34]:

```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier()
```

In [35]:

```
model.fit(X_train,y_train)
```

Out[35]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [36]:

```
#metrics to get performance
print('Accuracy',model.score(X_test,y_test)*100)
```

```
Accuracy 97.77777777777777
```

THANK YOU!

GitHub Link: https://github.com/anujtiwari21?tab=repositories (https://github.com/anujtiwari21?tab=repositories)