TASK 3 :- Iris Flower Classification

Author:- Md Nadim

Batch: January-2025

Domain: Data Science

```
In [99]:
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

```
In [5]:
df=pd.read_csv('C:/Users/Nadim/Downloads/archive (8)/IRIS.csv')
df.head()
```

Out[5]:

	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]:
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 [9]:
df.isnull().sum()
```

```
Out[9]:
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

```
In [11]:
df.describe()
```

Out[11]:

	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

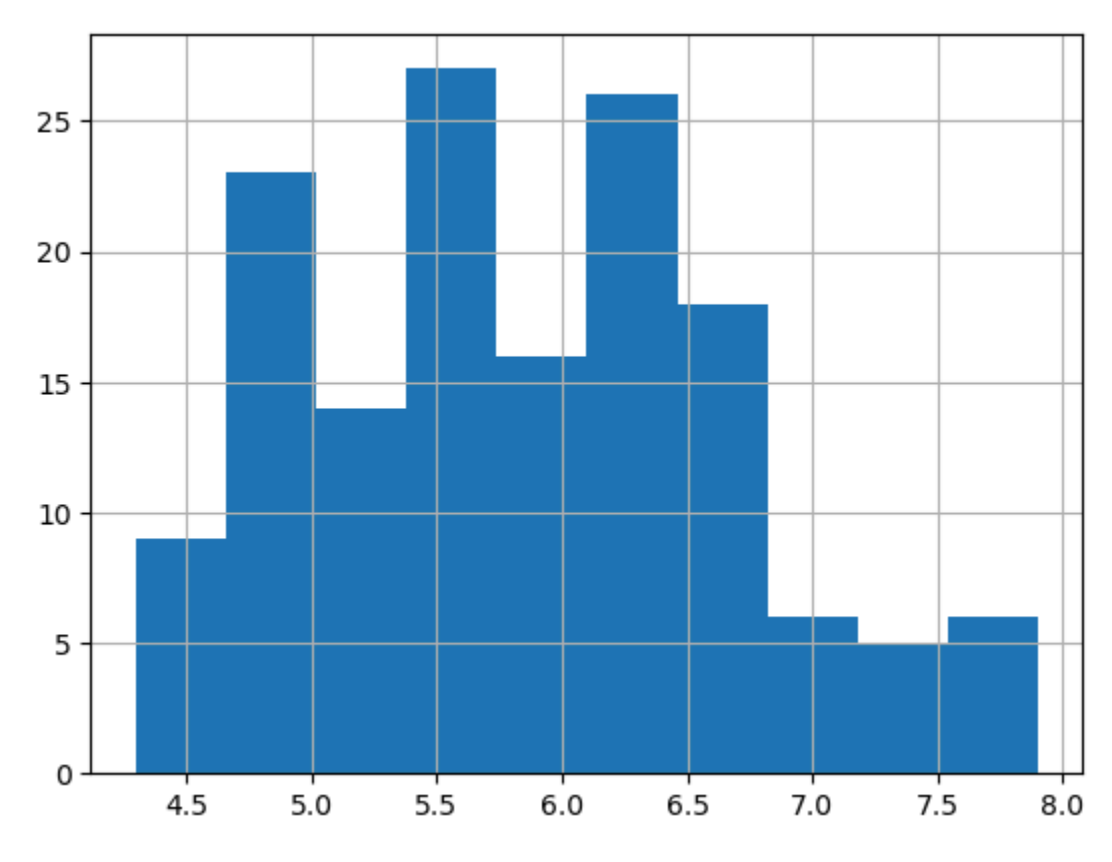
```
In [13]:
df['species'].value_counts()
```

```
Out[13]:
species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

Visual representation

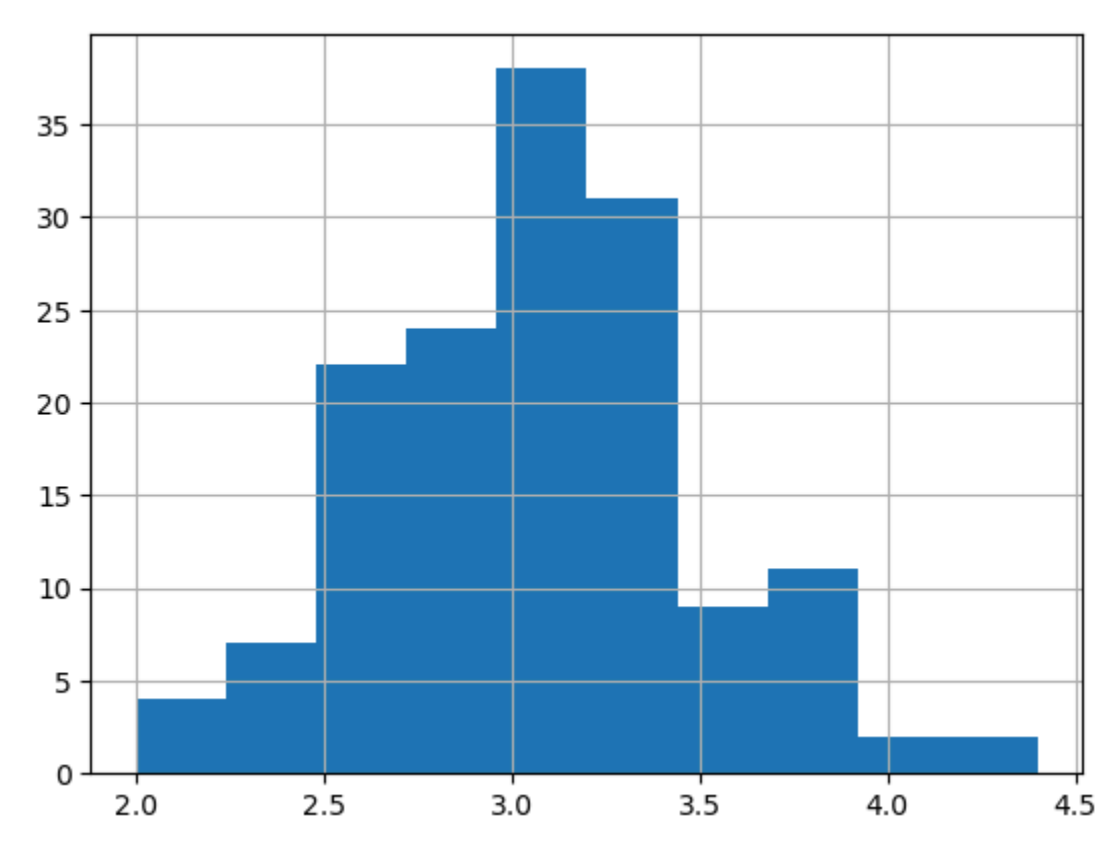
```
In [15]:
df['sepal_length'].hist()
```

```
Out[15]: <Axes: >
```



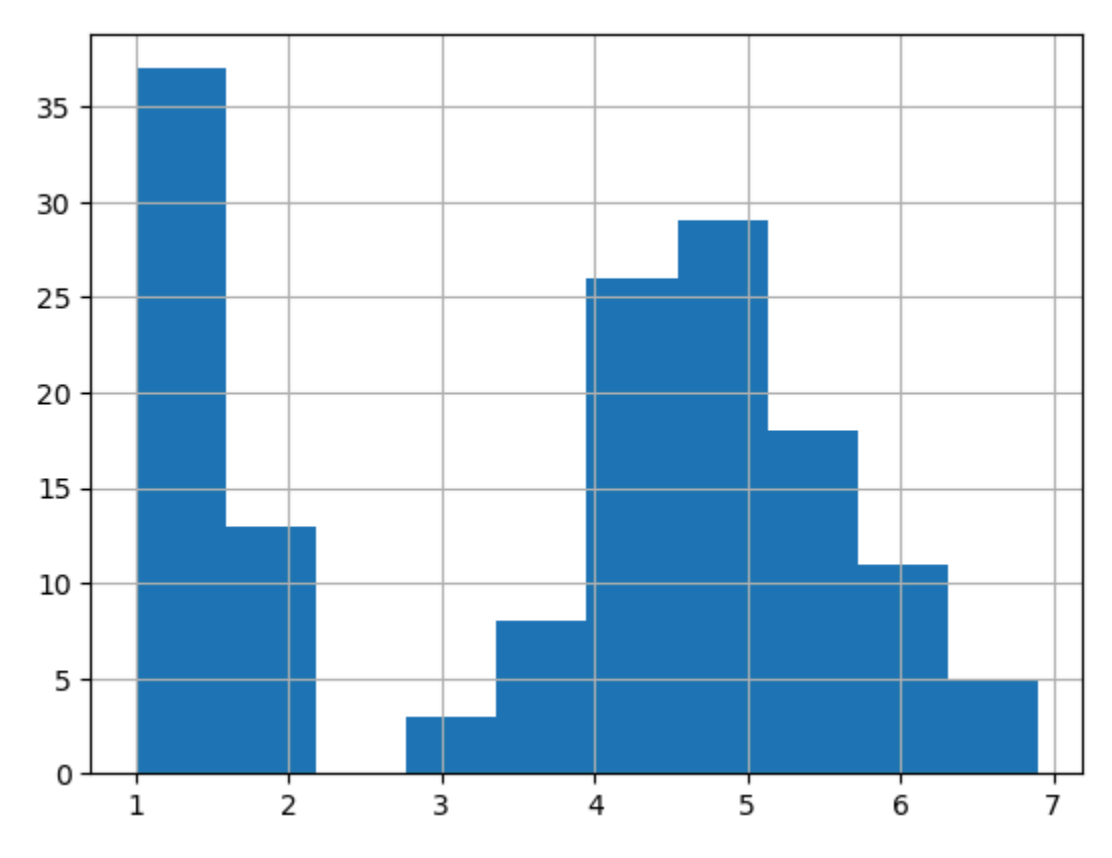
```
In [17]:
df['sepal_width'].hist()
```

```
Out[17]: <Axes: >
```



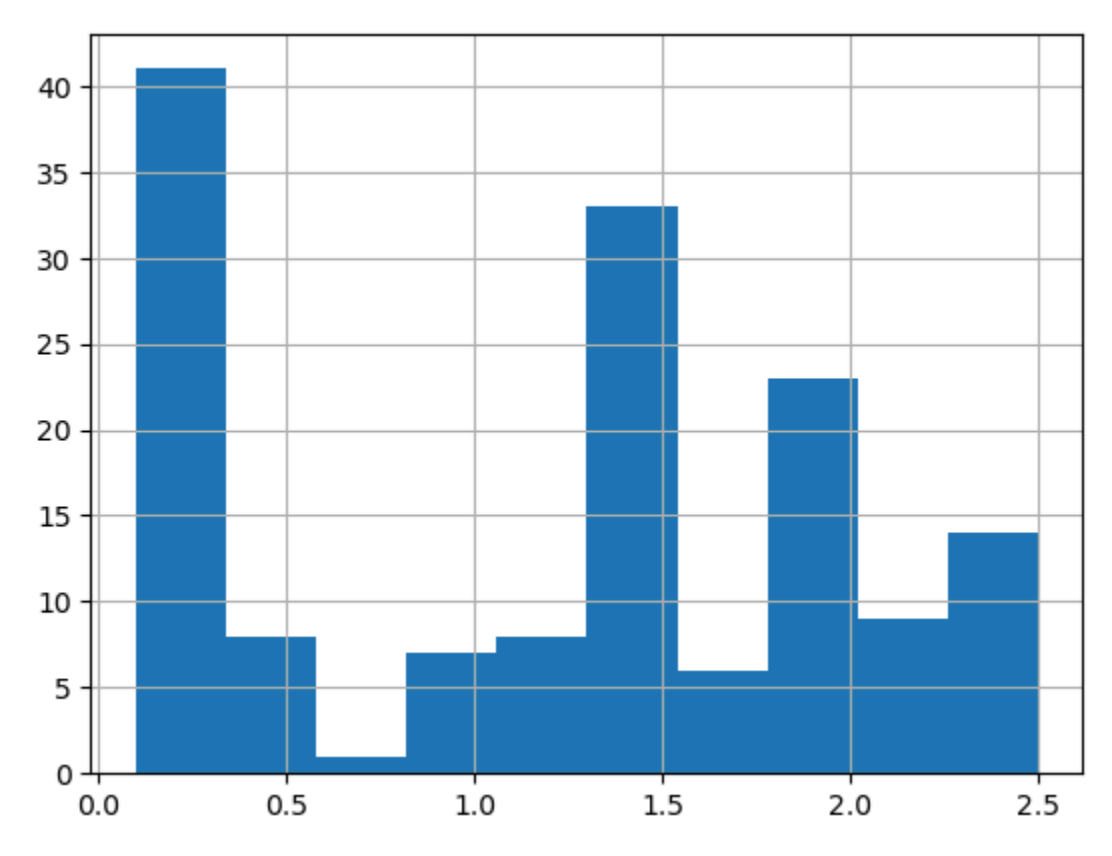
```
In [19]:
df['petal_length'].hist()
```

```
Out[19]: <Axes: >
```



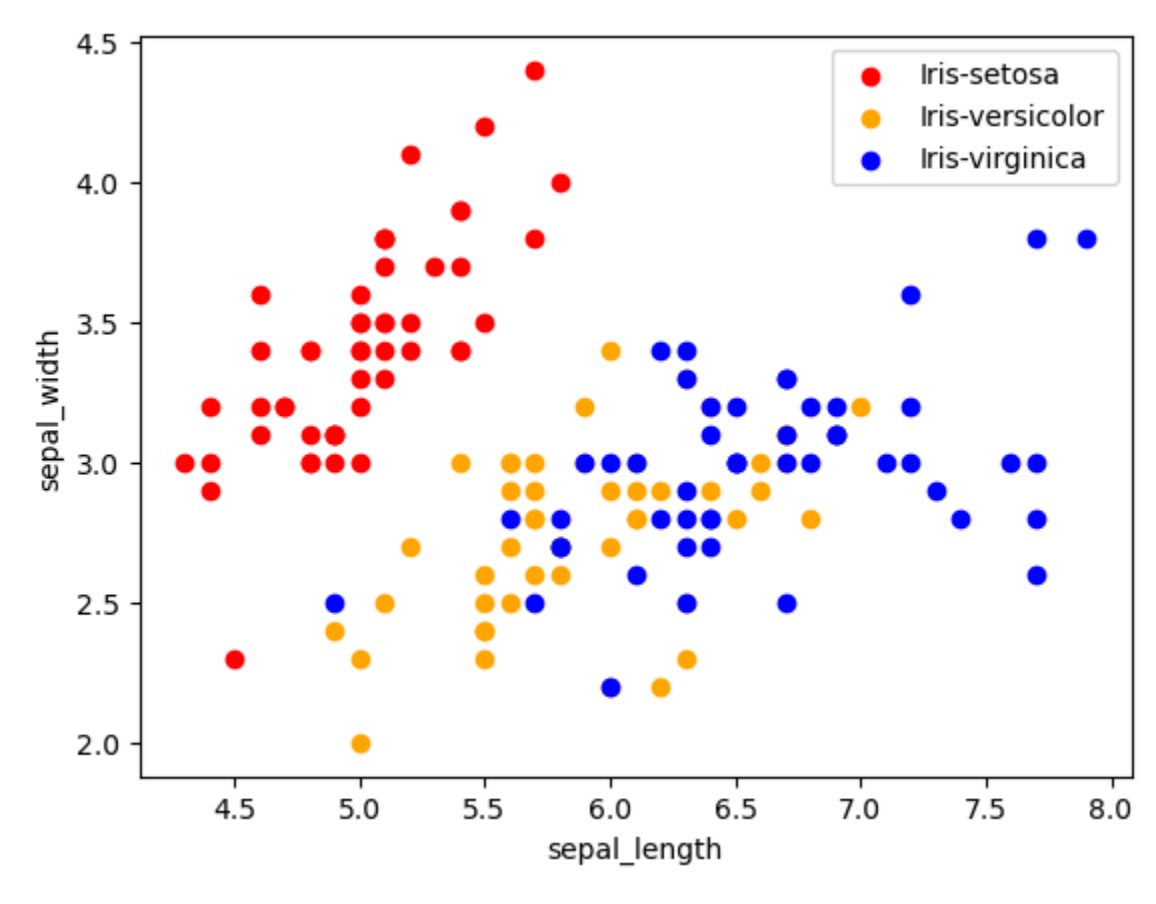
```
In [21]:
df['petal_width'].hist()
```

```
Out[21]: <Axes: >
```



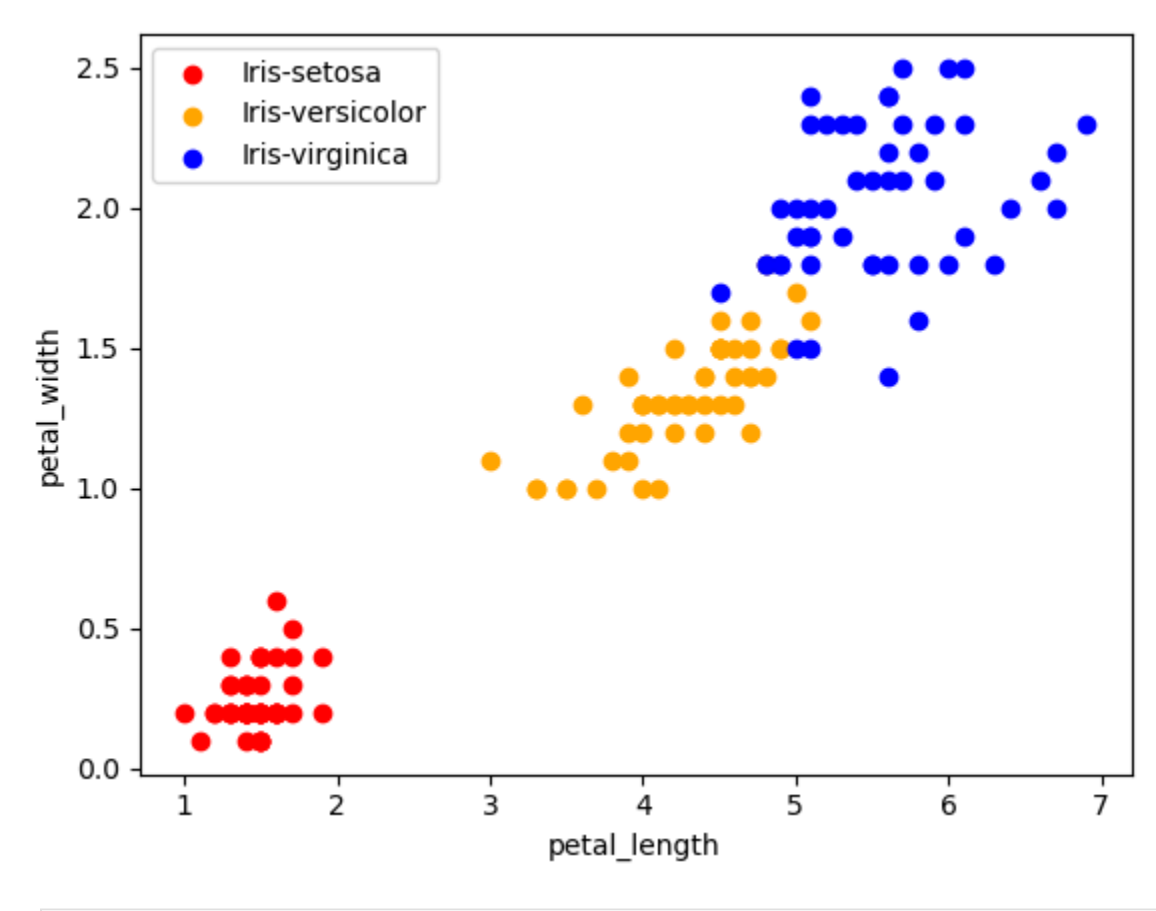
```
In [23]:
colors=['red','orange','blue']
species=['Iris-setosa','Iris-versicolor','Iris-virginica']

for i in range(3):
    x=df[df['species']==species[i]]
    plt.scatter(x['sepal_length'],x['sepal_width'],c=colors[i],label=species[i])
plt.xlabel('sepal_length')
plt.ylabel('sepal_width')
plt.legend()
```

```
Out[23]: <matplotlib.legend.Legend at 0x1c565d13f50>
```



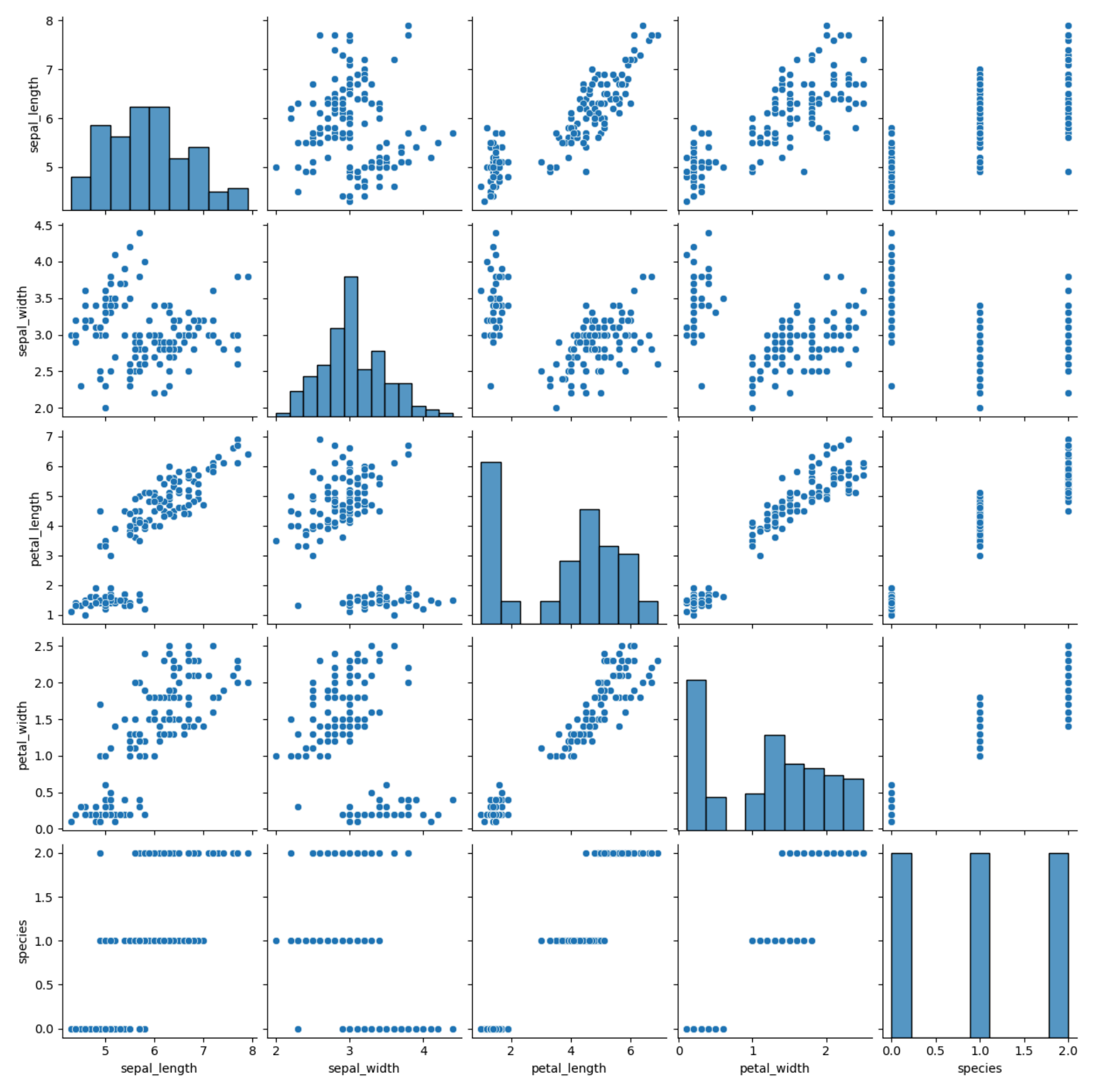

```
In [25]:
colors=['red','orange','blue']
species=['Iris-setosa','Iris-versicolor','Iris-virginica']

for i in range(3):
    x=df[df['species']==species[i]]
    plt.scatter(x['petal_length'],x['petal_width'],c=colors[i],label=species[i])
plt.xlabel('petal_length')
plt.ylabel('petal_width')
plt.legend()
```

```
Out[25]: <matplotlib.legend.Legend at 0x1c56676ed80>
```




```
In [103…
sns.pairplot(df)
```

```
Out[103… <seaborn.axisgrid.PairGrid at 0x1c56d937590>
```

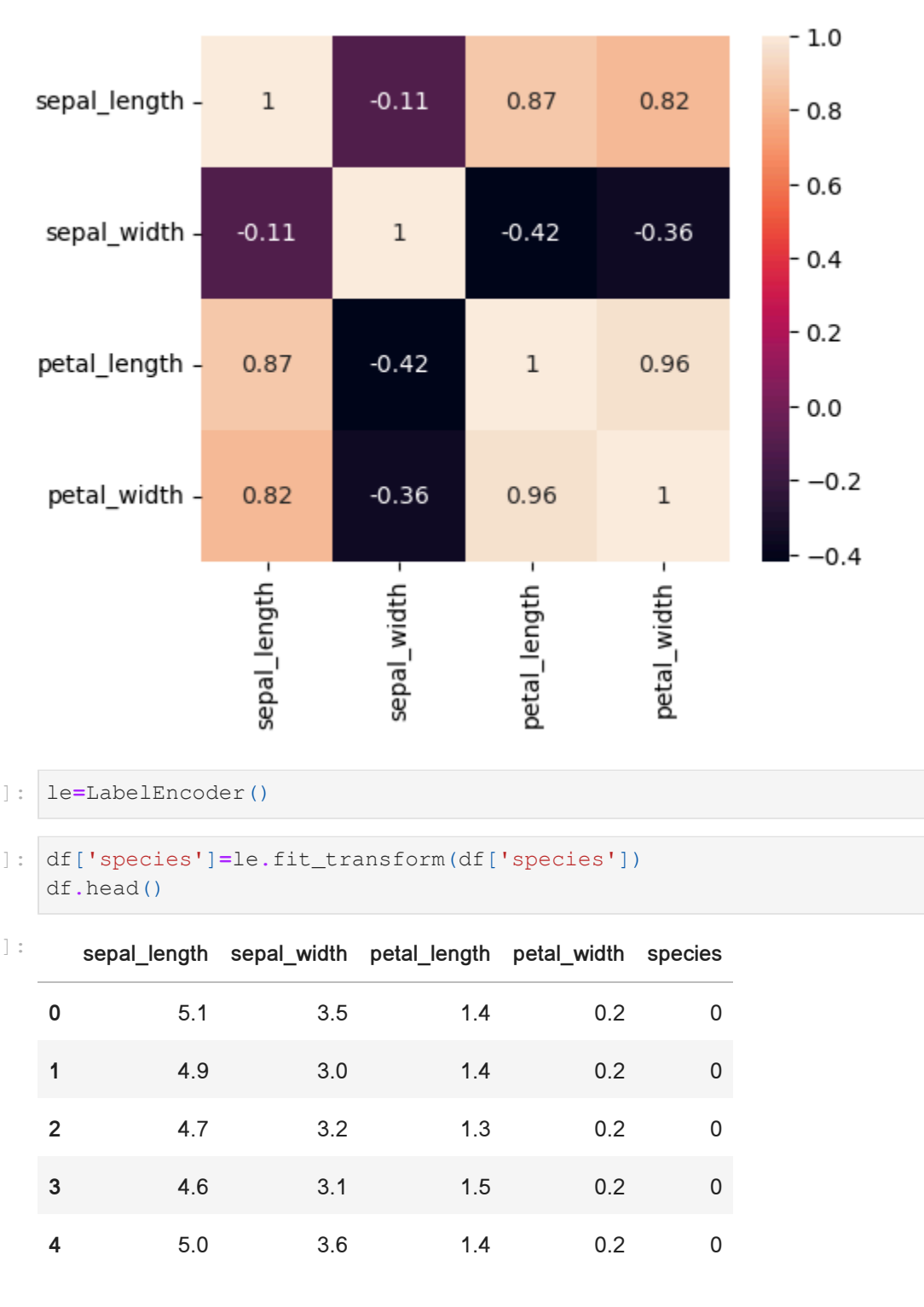



```
In [105…
numerical_data = df.select_dtypes(include=['float64', 'int64'])

print(numerical_data.corr())
```

```
              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 [64]:
Corr=numerical_data.corr()
fig, ax=plt.subplots(figsize=(5,4))
sns.heatmap(Corr, annot=True, ax=ax)
```

```
Out[64]: <Axes: >
```




```
In [66]:
le=LabelEncoder()
```

```
In [68]:
df['species']=le.fit_transform(df['species'])
df.head()
```

Out[68]:

	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

Model training

```
In [70]:
X=df.drop(columns=['species'])
Y=df['species']
```

```
In [72]:
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2, random_state=2)
```

```
In [74]:
print(X.shape,X_train.shape,X_test.shape)
```

```
(150, 4) (120, 4) (30, 4)
```

```
In [76]:
model=KNeighborsClassifier()
```

```
In [78]:
model.fit(X_train, Y_train)
```

```
Out[78]:
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
In [80]:
# accuracy on training data

X_train_prediction=model.predict(X_train)
```

```
In [82]:
print(X_train_prediction)
```

```
[2 0 1 2 1 0 2 1 1 2 1 1 2 1 0 2 0 1 0 0 0 1 2 2 0 2 2 2 1 0 0 2 1 1 2 2 1
 0 1 0 2 1 1 0 1 1 1 2 0 1 0 1 2 0 1 0 0 0 2 2 0 0 2 2 1 2 1 1 2 0 2 2 2 0
 2 0 0 1 2 1 2 1 1 2 1 1 1 2 2 2 2 0 1 1 1 1 2 1 0 0 2 1 2 0 2 0 2 2 0 1 0
 2 1 0 2 1 0 0 2 0]
```

```
In [84]:
training_data_accuracy =accuracy_score(Y_train, X_train_prediction)

print('Accuracy score of training data :', training_data_accuracy )
```

```
Accuracy score of training data : 0.9583333333333334
```

```
In [86]:
# Accuracy on test data

X_test_prediction =model.predict(X_test)
```

```
In [88]:
print(X_test_prediction)
```

```
[0 0 2 0 0 2 0 2 2 0 0 0 0 0 1 1 0 1 2 1 1 1 2 1 1 0 0 2 0 2]
```

```
In [90]:
testing_data_accuracy=accuracy_score(Y_test,X_test_prediction)

print('Accuracy score of testing data :', testing_data_accuracy)
```

```
Accuracy score of testing data : 1.0
```

Thank You

```
In [ ]:
```