Iris Dataset Analysis - Classification

May 15, 2024

1 Iris Dataset Analysis - Classification

[1]:
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

[2]:
```
df = pd.read_csv(r"C:\Users\Ricky\Downloads\IRIS.csv")
```

[3]:
```
df.head()
```

[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

[4]:
```
# to display stats about data
df.describe()
```

[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

[5]:
```
# to basic info about datatype
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
```

[7]:
```
# to display no. of samples on each class
df['species'].value_counts()
```

[7]:
```
species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

2 Preprocessing the dataset

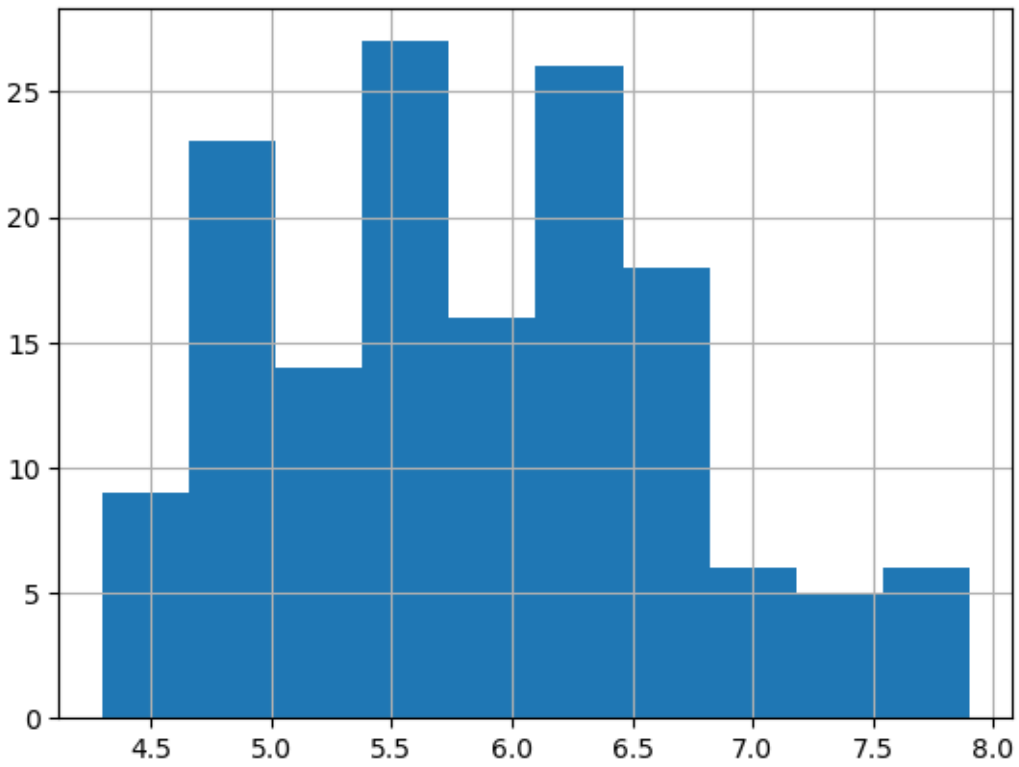
[9]:
```
# check for null values
df.isnull().sum()
```

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

3 Exploratory Data Analysis

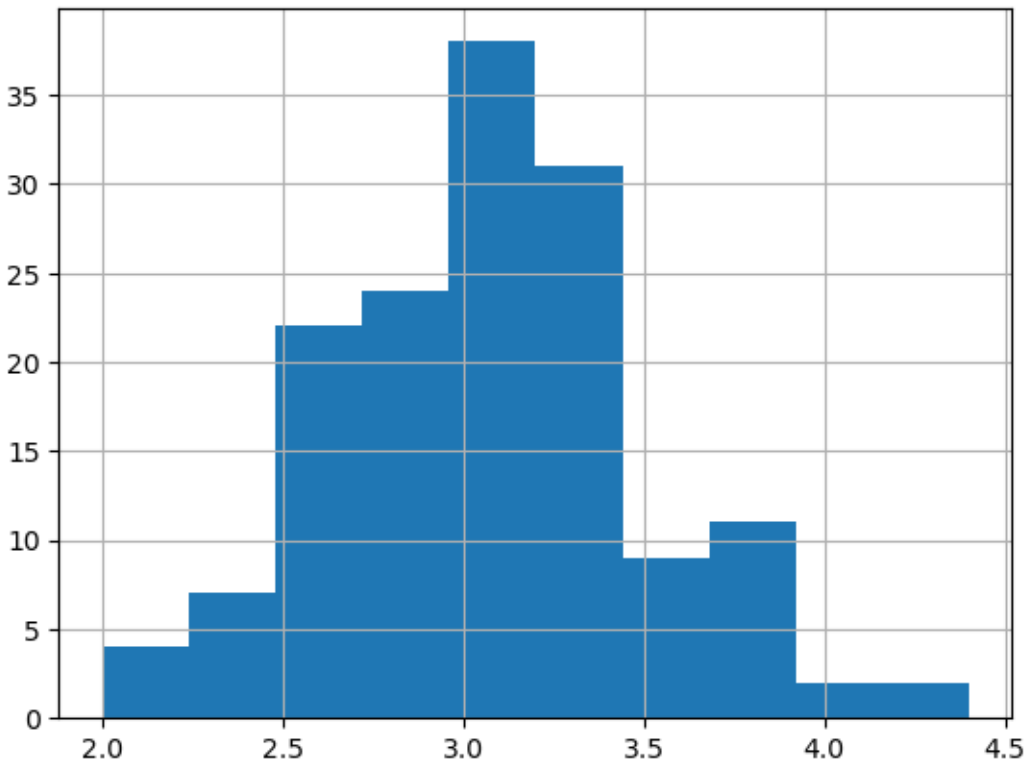
[11]:
```
# histograms
df['sepal_length'].hist()
```

[11]:
```
<Axes: >
```



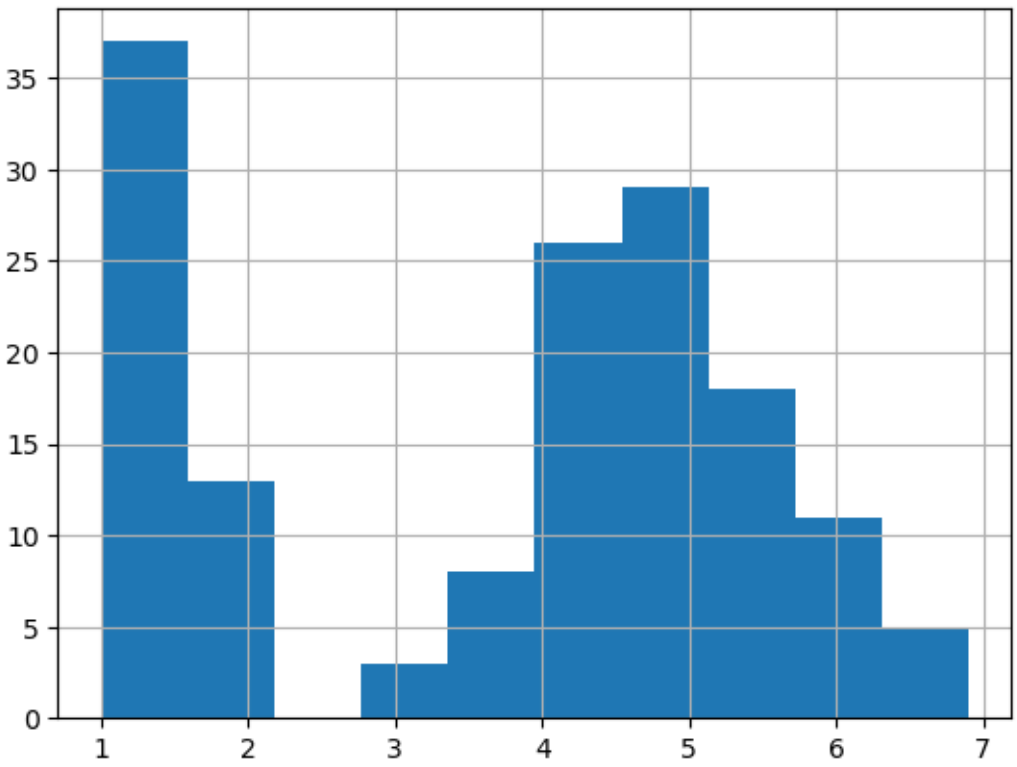
[13]:
```
# histograms
df['sepal_width'].hist()
```

[13]: <Axes: >



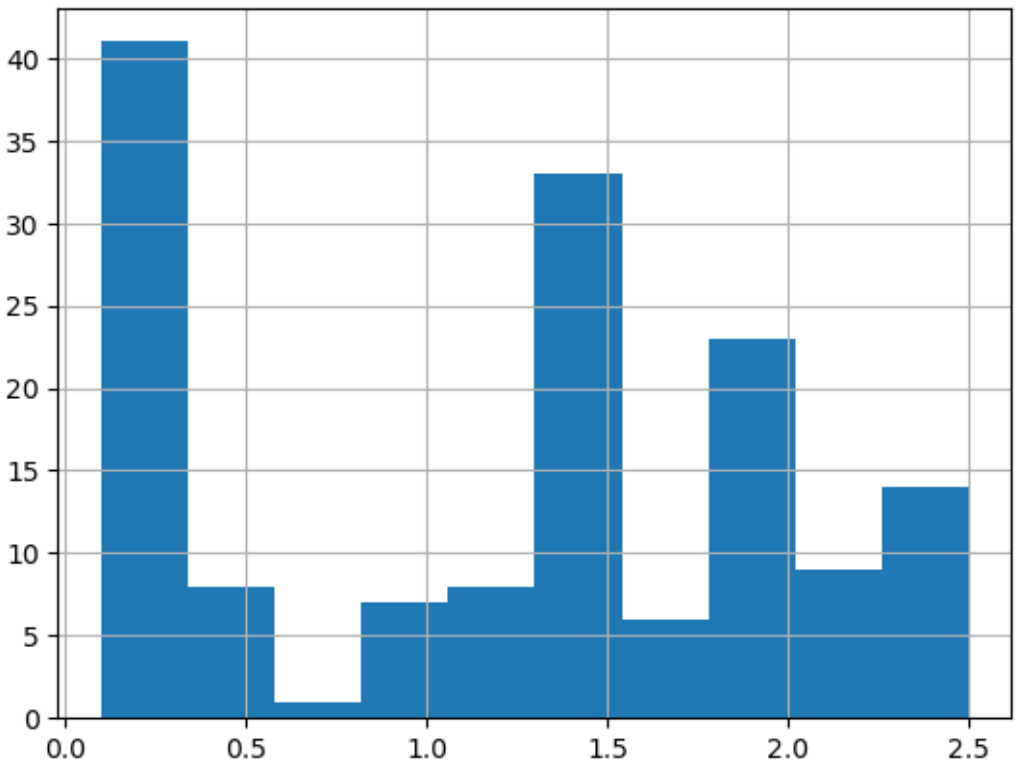
[14]:
```
df['petal_length'].hist()
```

[14]: <Axes: >



[15]:
```
df['petal_width'].hist()
```

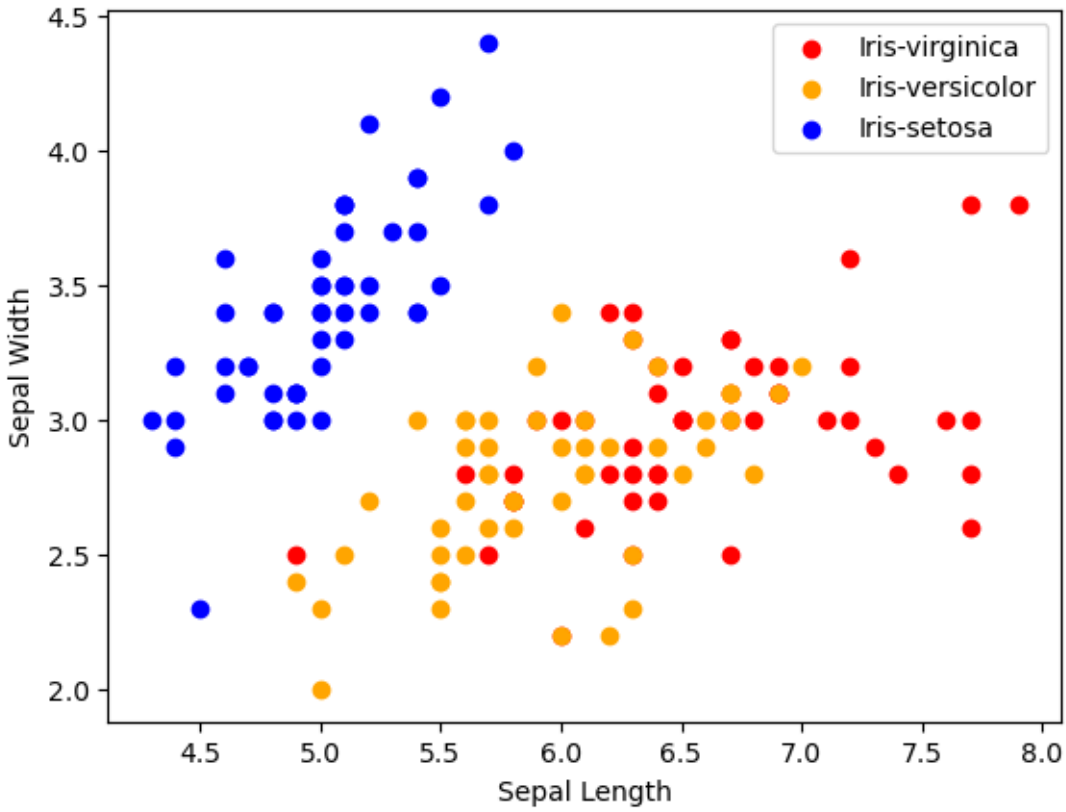
[15]: <Axes: >



[16]:
```
# scatterplot
colors = ['red', 'orange', 'blue']
species = ['Iris-virginica','Iris-versicolor','Iris-setosa']
```

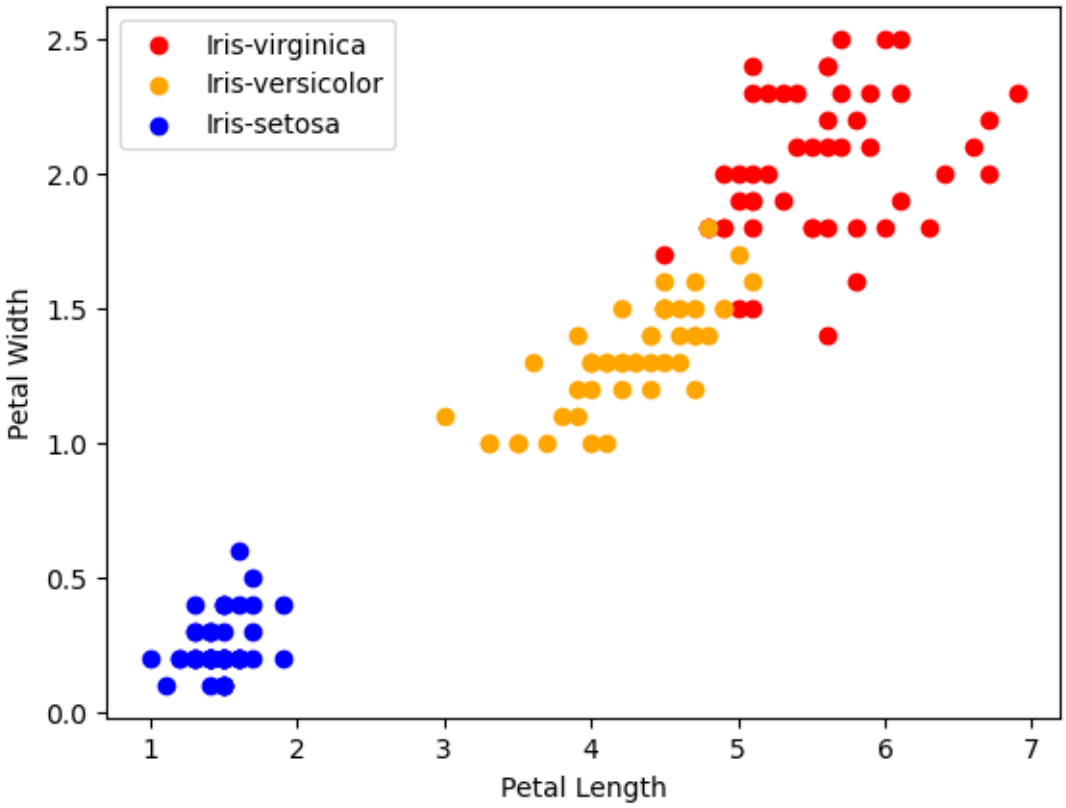
[18]:
```
for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['sepal_length'], x['sepal_width'], c = colors[i],␣
 ↪label=species[i])
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.legend()
```

[18]:
```
<matplotlib.legend.Legend at 0x227577ad390>
```



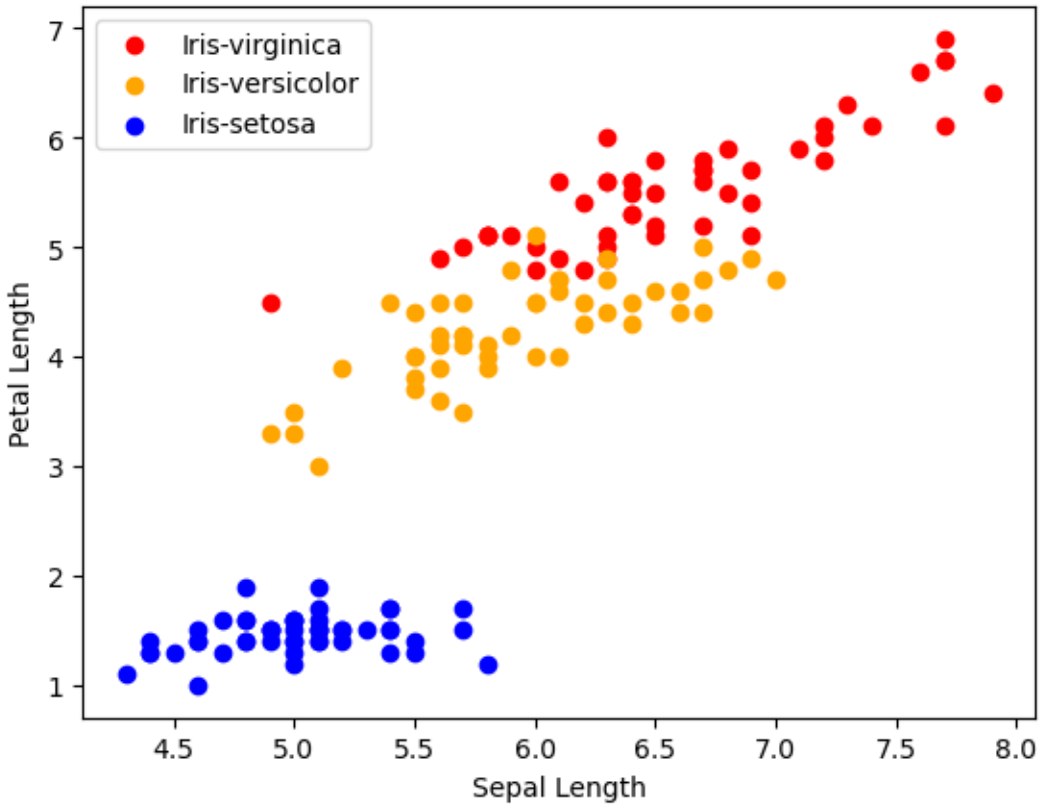
[22]:
```
for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['petal_length'], x['petal_width'], c = colors[i],␣
 ↪label=species[i])
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
plt.legend()
```

[22]: <matplotlib.legend.Legend at 0x22757ff7f10>



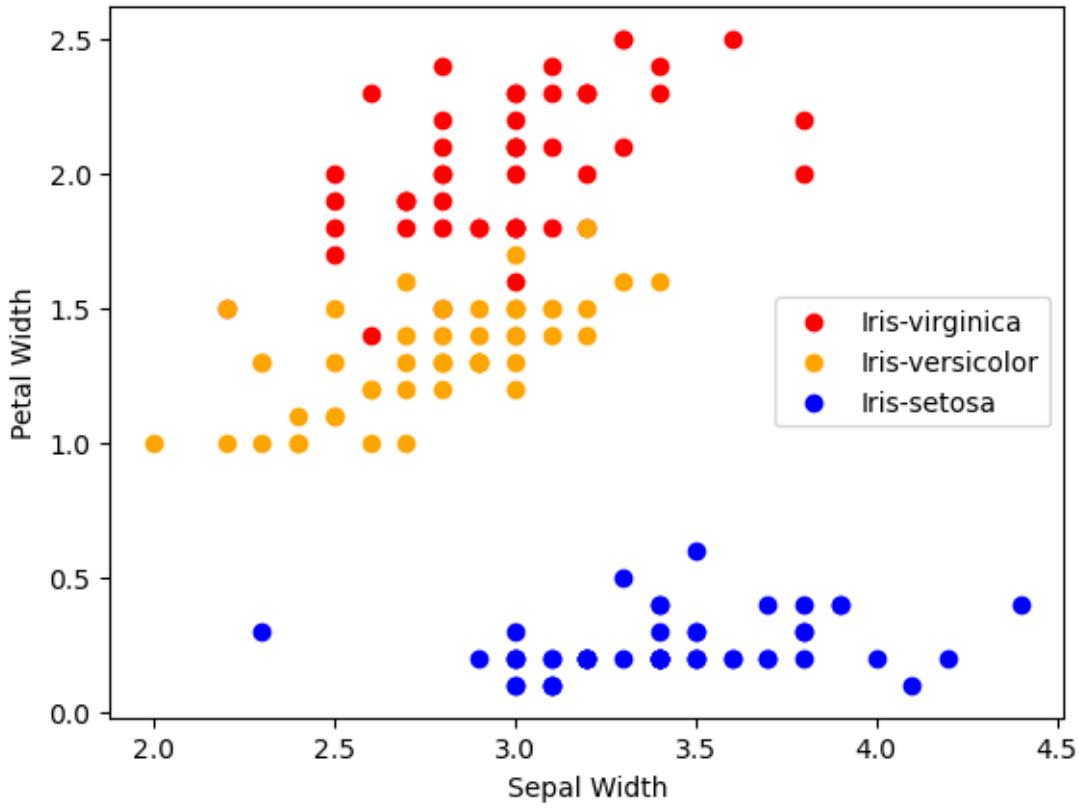
[23]:
```
for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['sepal_length'], x['petal_length'], c = colors[i],␣
 ↪label=species[i])
plt.xlabel("Sepal Length")
plt.ylabel("Petal Length")
plt.legend()
```

[23]: <matplotlib.legend.Legend at 0x2275824d190>



[24]:
```
for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['sepal_width'], x['petal_width'], c = colors[i],␣
 ↪label=species[i])
plt.xlabel("Sepal Width")
plt.ylabel("Petal Width")
plt.legend()
```

[24]: <matplotlib.legend.Legend at 0x22757f77f10>

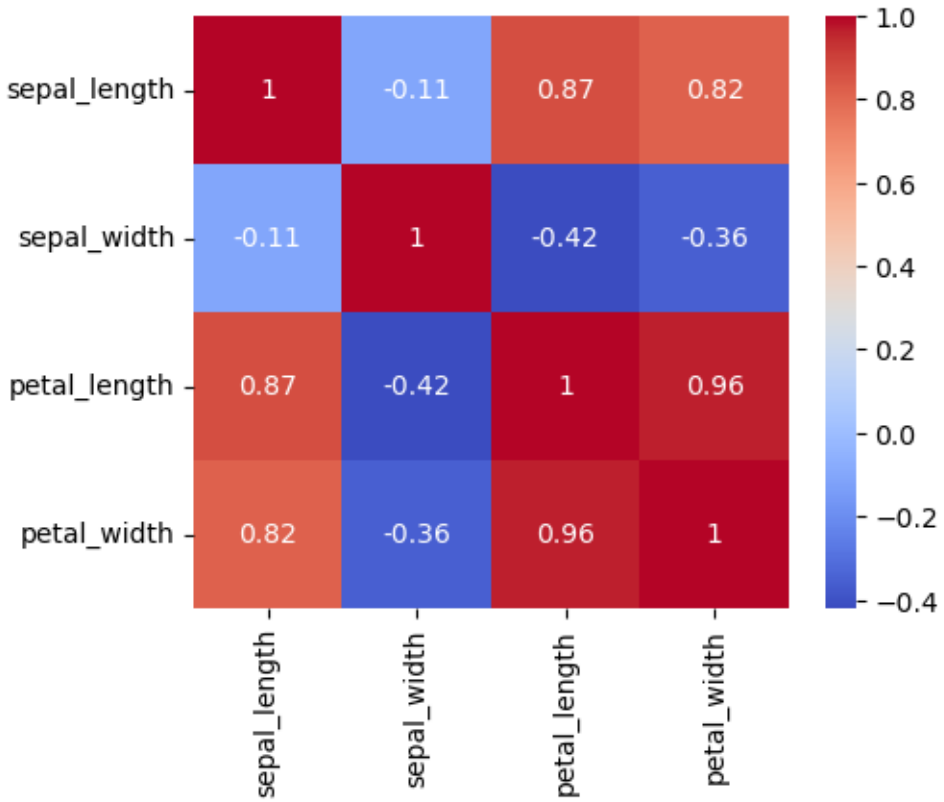


4 Coorelation Matrix

[27]:
```
numeric_df = df.drop(columns=['species']).select_dtypes(include=['float64',␣
↪'int64'])
correlation_matrix = numeric_df.corr()
```

[28]:
```
corr = numeric_df.corr()
fig, ax = plt.subplots(figsize=(5,4))
sns.heatmap(corr, annot=True, ax=ax, cmap = 'coolwarm')
```

[28]: <Axes: >



5 Label Encoder

[29]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

[31]:
```
df['species'] = le.fit_transform(df['species'])
df.head()
```

[31]:

	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

6 Model Training

[32]:
```
from sklearn.model_selection import train_test_split
# train - 70
# test - 30
X = df.drop(columns=['species'])
Y = df['species']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.30)
```

[33]:
```
# logistic regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

[34]:
```
# model training
model.fit(x_train, y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[34]:
```
LogisticRegression()
```

[35]:
```
# print metric to get performance
print("Accuracy: ",model.score(x_test, y_test) * 100)
```

```
Accuracy:  95.55555555555556
```

[36]:
```
# knn - k-nearest neighbours
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
```

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

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

[38]:
```
# print metric to get performance
print("Accuracy: ",model.score(x_test, y_test) * 100)
```

```
Accuracy:  95.55555555555556
```

[ ]:
```

```