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

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

In [3]:
```
df.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]:
```
df.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]:
```
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 [6]:
```
df['species'].value_counts()
```

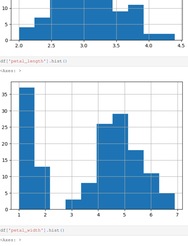
Out[6]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: species, dtype: int64
```

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

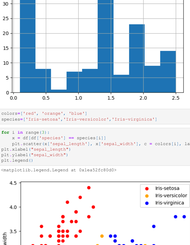
In [8]:
```
df['sepal_length'].hist()
```

Out[8]:
```
<Axes: >
```



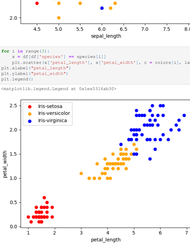
In [9]:
```
df['sepal_width'].hist()
```

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



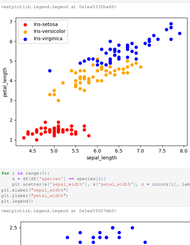
In [10]:
```
df['petal_length'].hist()
```

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



In [11]:
```
df['petal_width'].hist()
```

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



In [12]:
```
colors=["red", "orange", "blue"]
species=["Iris-setosa","Iris-versicolor","Iris-virginica"]
```

In [13]:
```
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[13]:
```
<matplotlib.legend.Legend at 0x1ea52fc80d0>
```

In [14]:
```
for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['sepal_length'], x['petal_width'], c = colors[i], label=species[i])
plt.xlabel("sepal_length")
plt.ylabel("petal_width")
plt.legend()
```

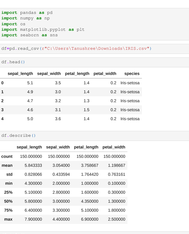
Out[14]:
```
<matplotlib.legend.Legend at 0x1ea531fab30>
```

In [15]:
```
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()
```

Out[15]:
```
<matplotlib.legend.Legend at 0x1ea533f8a60>
```

In [16]:
```
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()
```

Out[16]:
```
<matplotlib.legend.Legend at 0x1ea53007eb0>
```



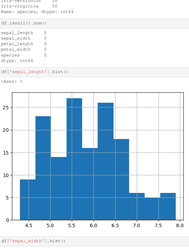
In [17]:
```
df.corr()
```

Out[17]:
```
C:\Users\Tanushree\AppData\Local\Temp\ipykernel_92445\1134702465.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  df.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 [18]:
```
corr = df.corr()
fig, ax = plt.subplots(figsize=(5,4))
sns.heatmap(corr, annot=True, ax=ax, cmap='coolwarm')
```

Out[18]:
```
C:\Users\Tanushree\AppData\Local\Temp\ipykernel_92445\2466531622.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  corr = df.corr()
<Axes: >
```



In [19]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [20]:
```
df["species"] = le.fit_transform(df["species"])
df.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]:
```
from sklearn.model_selection import train_test_split
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)
```

In [22]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [23]:
```
model.fit(x_train, y_train)
```

Out[23]:
```
* LogisticRegression
LogisticRegression()
```

In [24]:
```
print("Accuracy: ", model.score(x_test, y_test)* 100)
```

```
Accuracy:  97.77777777777777
```

In [25]:
```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier()
```

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

Out[26]:
```
* KNeighborsClassifier
KNeighborsClassifier()
```

In [27]:
```
print("Accuracy: ", model.score(x_test, y_test)* 100)
```

```
Accuracy:  97.77777777777777
```

In [28]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
```

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

Out[29]:
```
* DecisionTreeClassifier
DecisionTreeClassifier()
```

In [30]:
```
print("Accuracy: ", model.score(x_test, y_test)* 100)
```

```
Accuracy:  95.55555555555556
```