iris1

January 6, 2024

[9]:
```
#required libraries
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

[10]:
```
#load data
df = pd.read_csv('Iris.csv')
df.head()
```

[10]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

[11]:
```
# dele coum
df = df.drop(columns = ['Id'])
df.head()
```

[11]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	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

[12]:
```
#statastics
df.describe()
```

[12]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
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
```

[13]:
```
#display basic info of data
df.info
```

[13]:
```
<bound method DataFrame.info of      SepalLengthCm  SepalWidthCm  PetalLengthCm
PetalWidthCm         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 x 5 columns]>
```

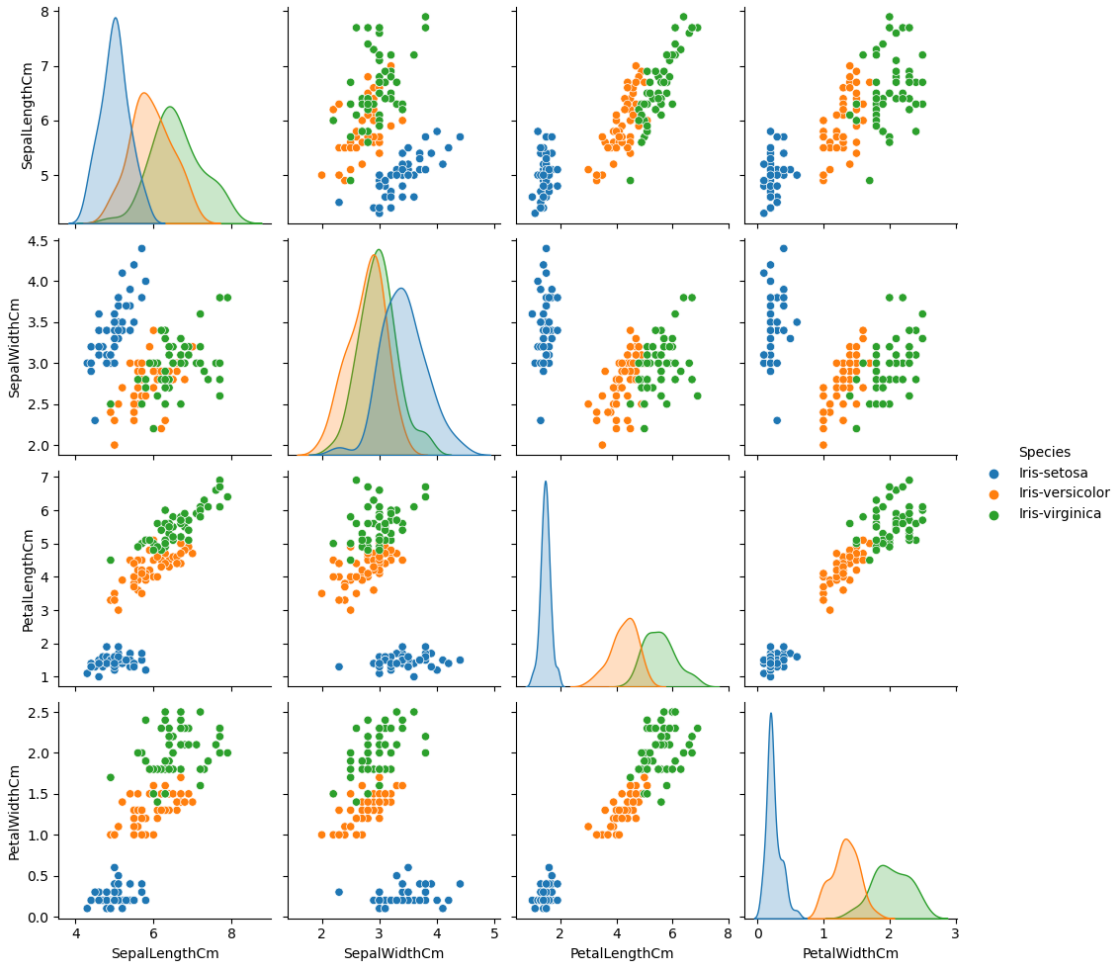
[14]:
```
#samole of each class
df['Species'].value_counts()
```

[14]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

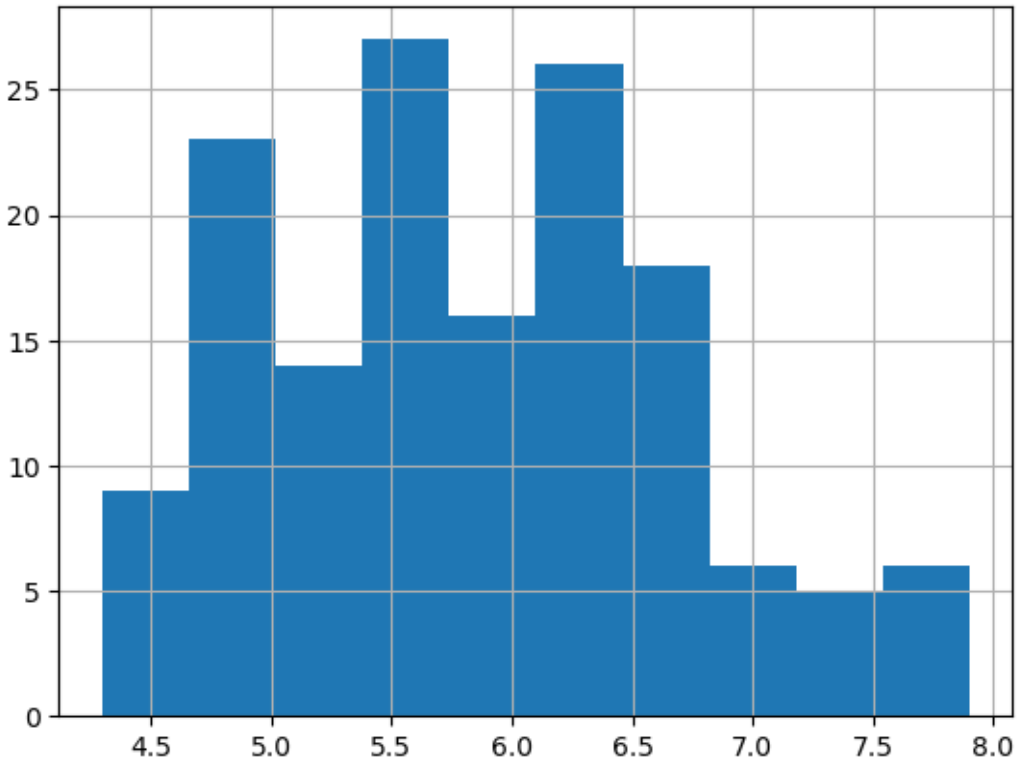
[15]:
```
#preprocess # Null Vlues
df.isnull().sum()
```

[15]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

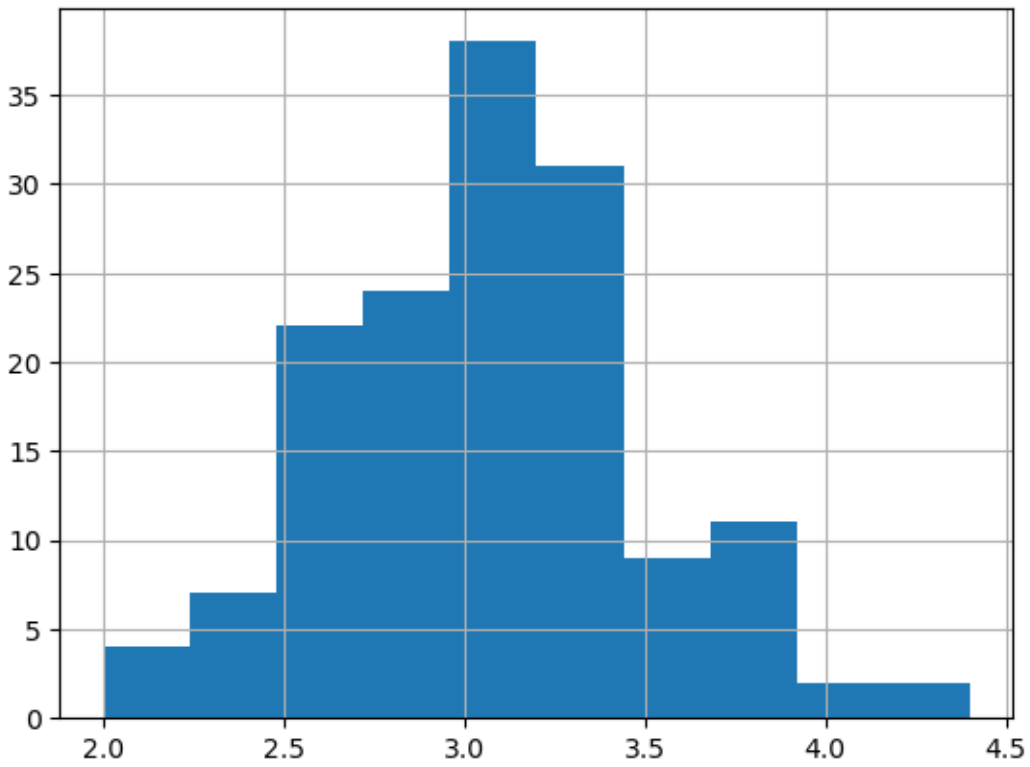
[16]:
```
sns.pairplot(df, hue='Species')
plt.show()
```



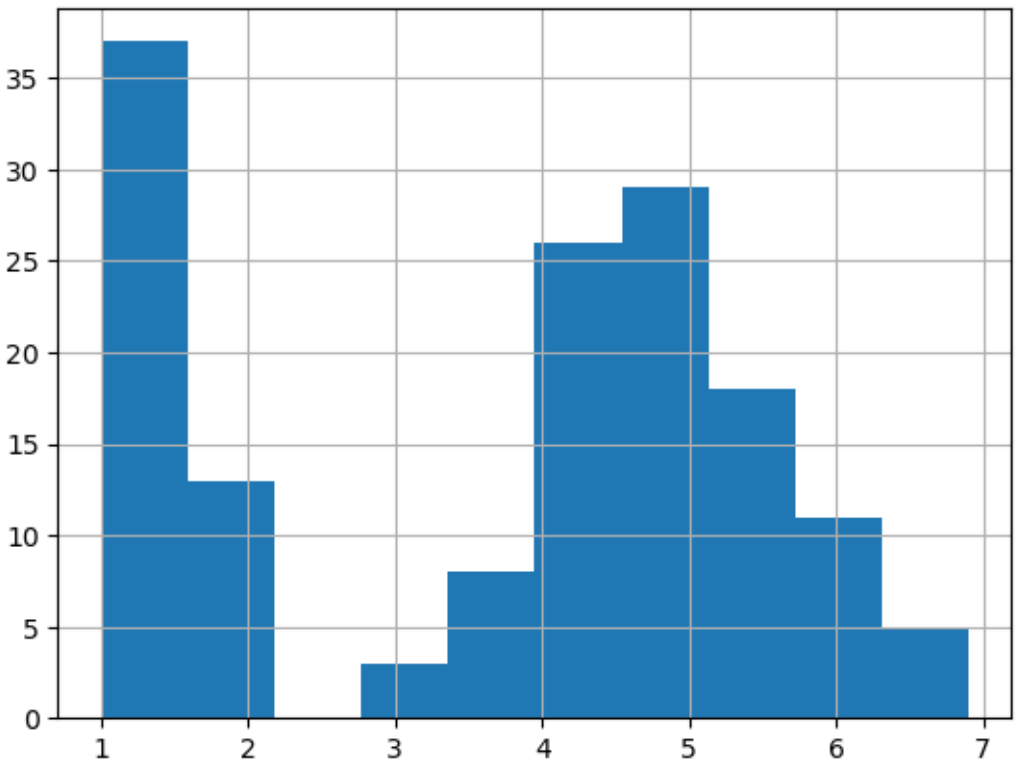
[17]:
```
#hisogram
df['SepalLengthCm'].hist()
plt.show()
```



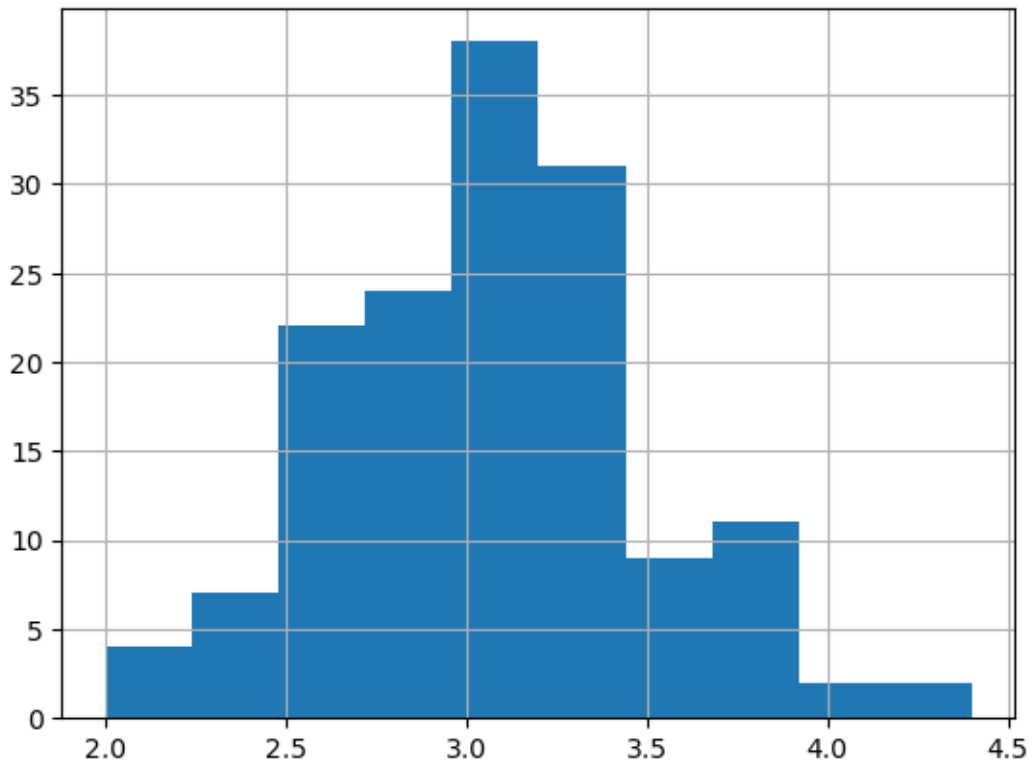
[18]:
```
#Histogram
df['SepalWidthCm'].hist()
plt.show()
```



[19]:
```
#Histogram
df['PetalLengthCm'].hist()
plt.show()
```



[20]:
```
#Histogram
df['SepalWidthCm'].hist()
plt.show()
```



[21]:
```
#Scatterplot
colors=['red','orange','blue']
Species = ['Iris-setosa','Iris-versicolor','Iris-virginica']
```

[4]:
```
```

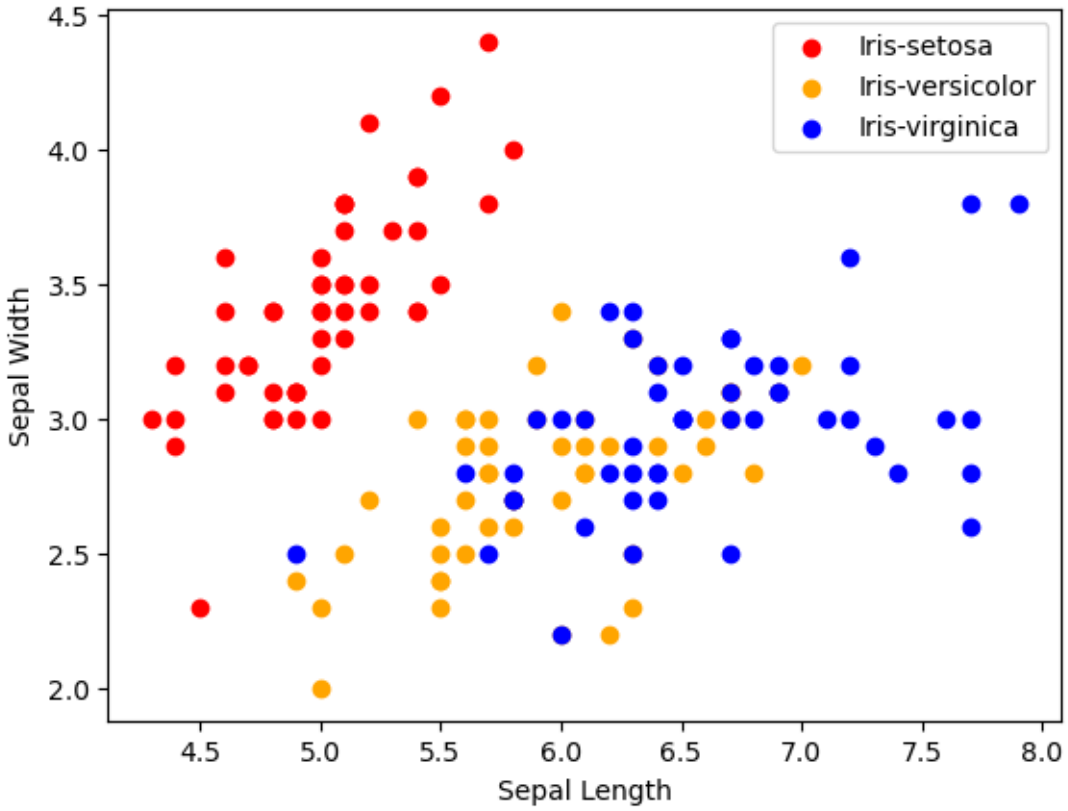
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_11020\43558000.py in <module>
      1 for i in range(3):
----> 2     x=df[df['Species']==Species[i]]
      3     plt.scatter(x['SepalLengthCm'],x['SepalWidthCm'], c = colors[i],␣
 ↪label=Species[i])
      4 plt.xlabel('Sepal Length')
      5 plt.xlabel('Sepal Width')

NameError: name 'df' is not defined
```

[ ]:
```
```

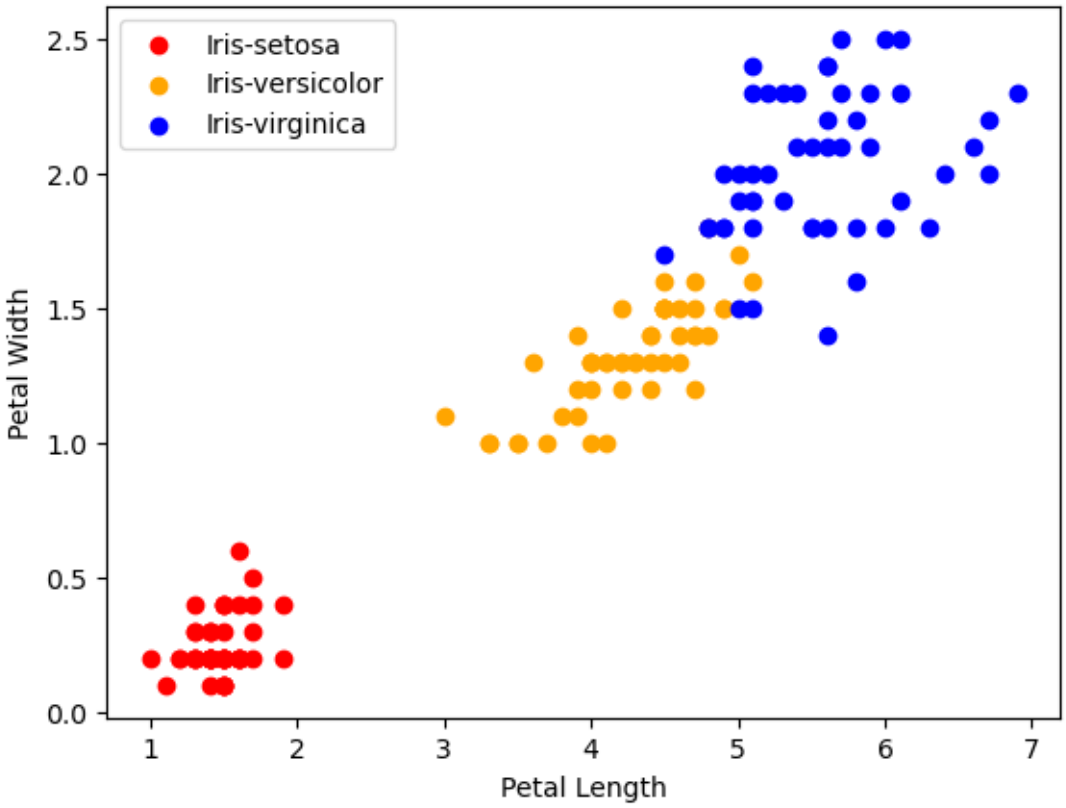
[22]:
```
#scatterplot
for i in range(3):
    x = df[df['Species'] == Species[i]]
    plt.scatter(x['SepalLengthCm'], x['SepalWidthCm'], c=colors[i],␣
 ↪label=Species[i])

plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.legend()
plt.show()
```



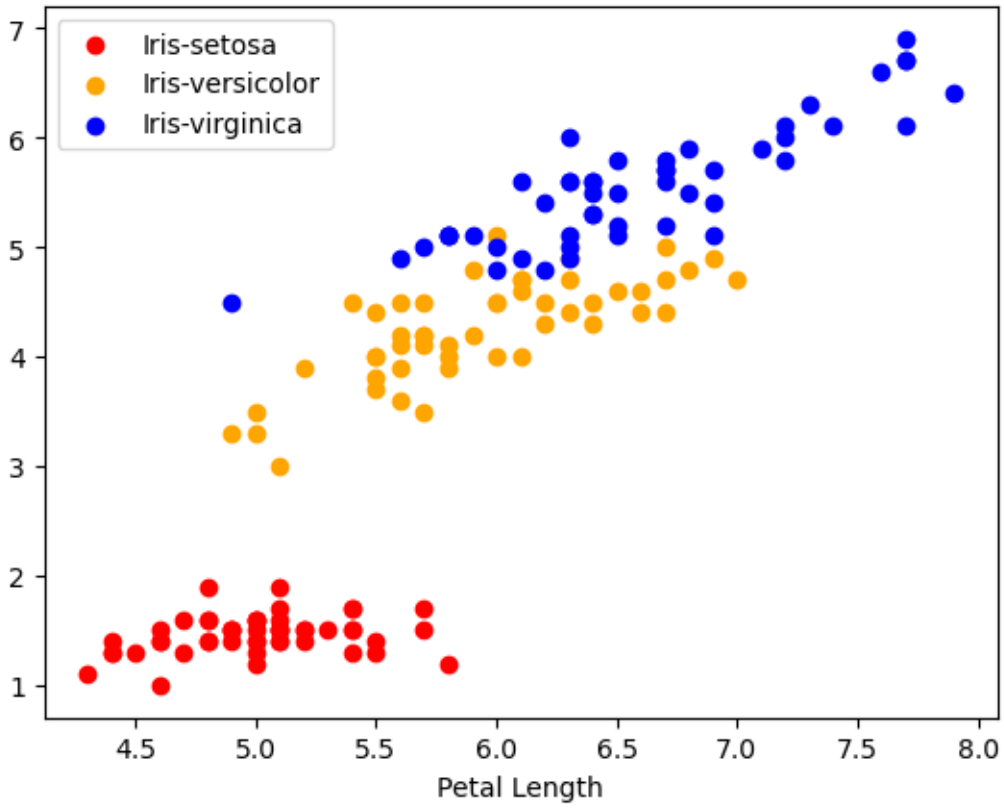
[23]:
```
#scatterplot
for i in range(3):
    x = df[df['Species'] == Species[i]]
    plt.scatter(x['PetalLengthCm'], x['PetalWidthCm'], c=colors[i],␣
 ↪label=Species[i])

plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
plt.legend()
plt.show()
```



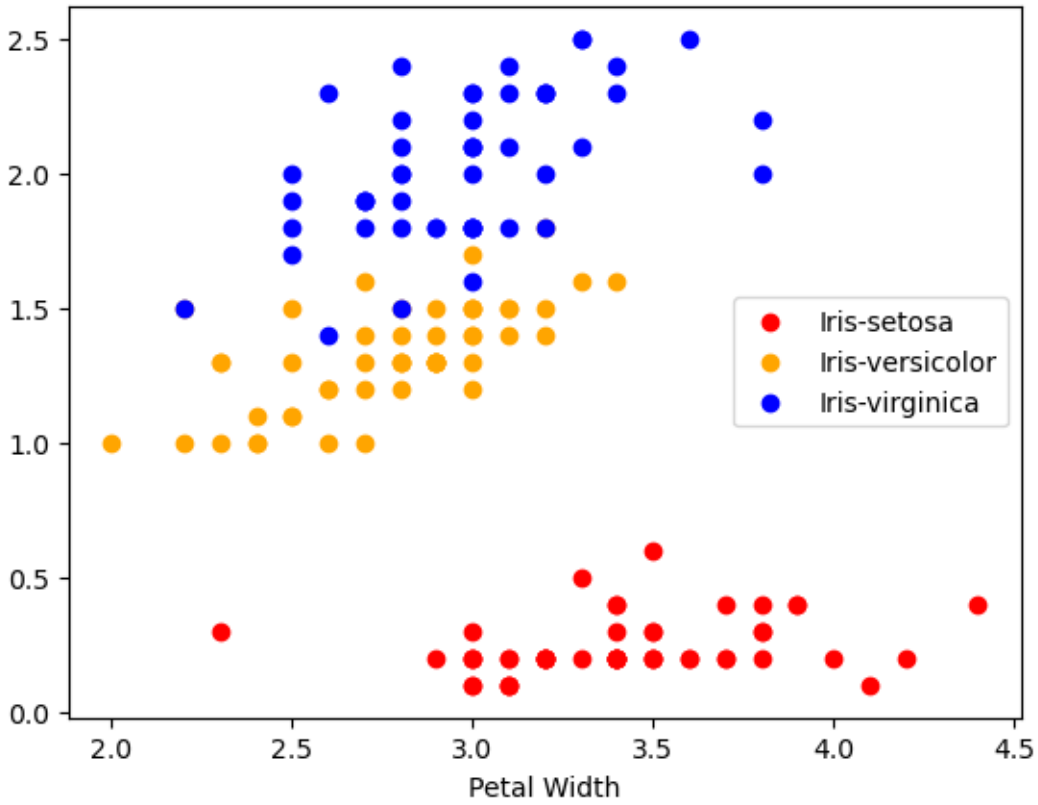
[24]:
```
#Scatterplot
for i in range(3):
    x = df[df['Species']==Species[i]]
    plt.scatter(x['SepalLengthCm'],x['PetalLengthCm'], c = colors[i],␣
 ↪label=Species[i])

plt.xlabel('Petal Length')
plt.xlabel('Petal Length')
plt.legend()
plt.show()
```

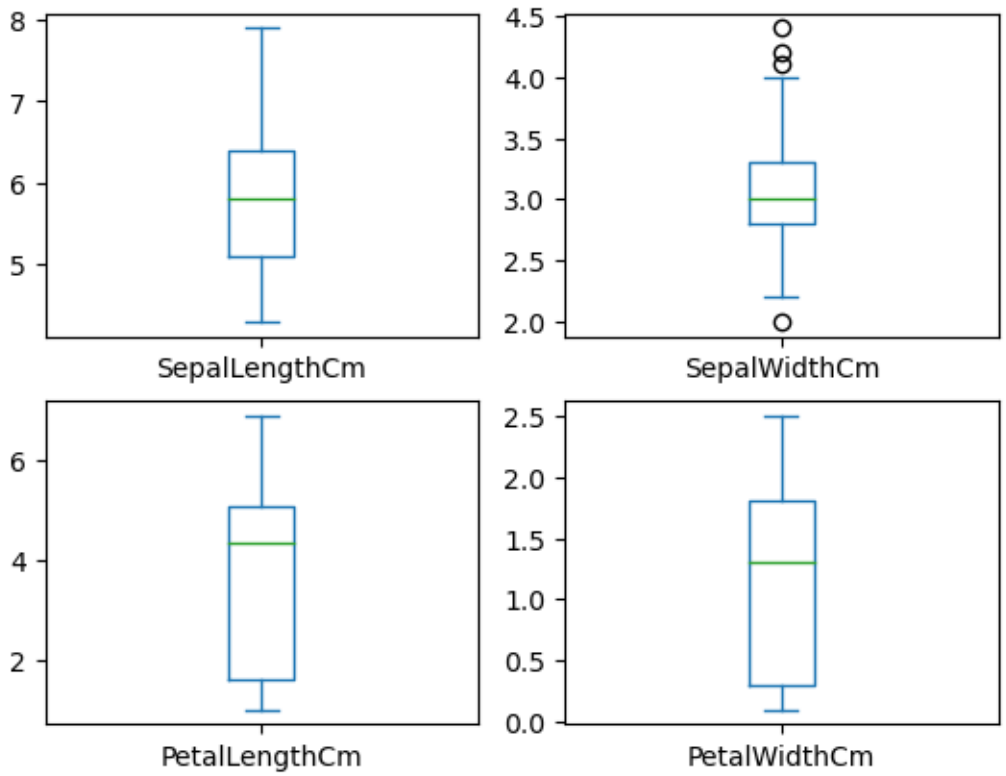


[25]:
```
#Scatterplot
for i in range(3):
    x = df[df['Species']==Species[i]]
    plt.scatter(x['SepalWidthCm'],x['PetalWidthCm'], c = colors[i],␣
 ↪label=Species[i])

plt.xlabel('Sepal Width')
plt.xlabel('Petal Width')
plt.legend()
plt.show()
```



[26]:
```
#Box plot
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```

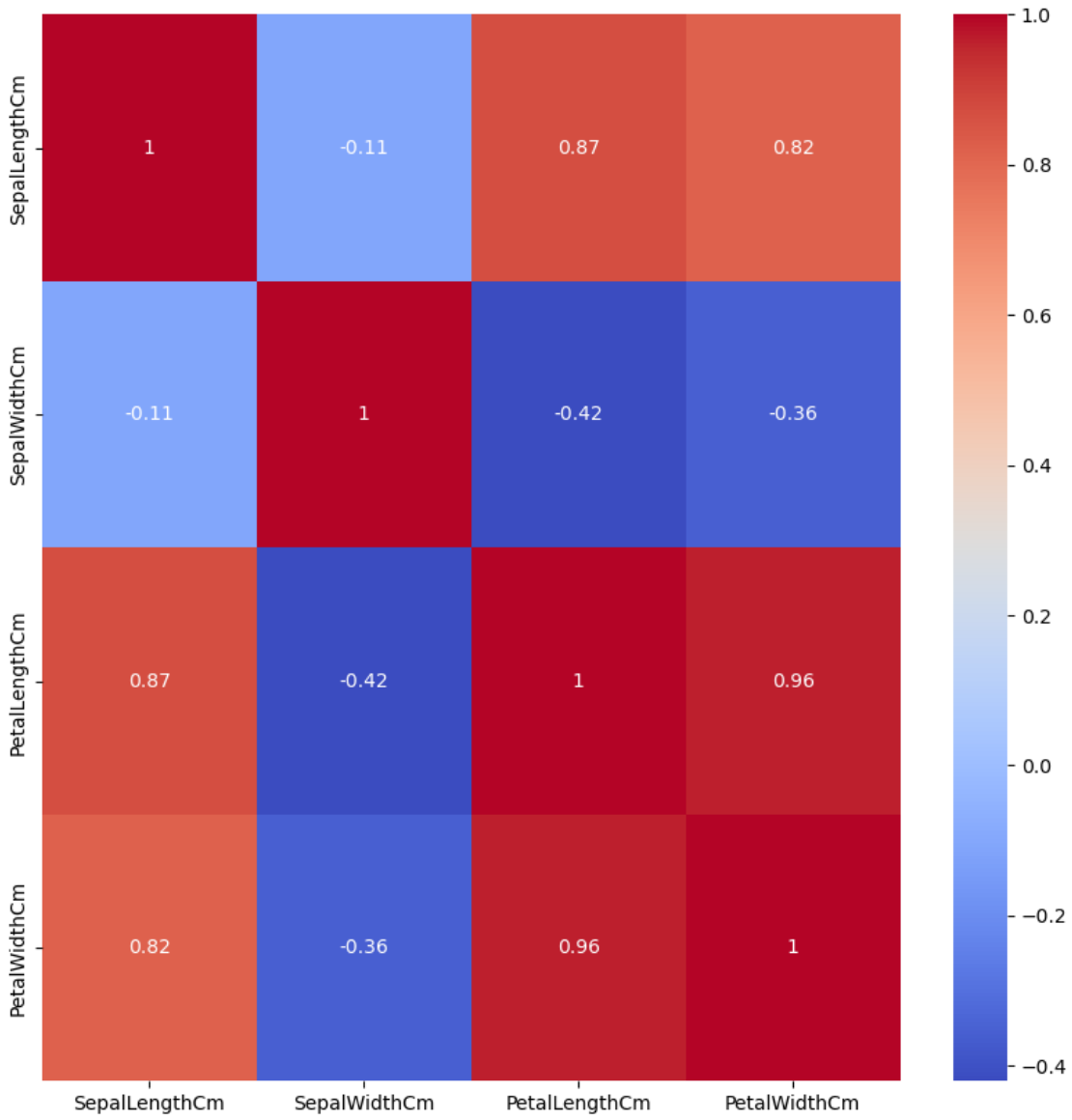


[27]:
```
df.corr()
```

[27]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

[30]:
```
corr = df.corr()
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(corr ,annot=True, ax=ax, cmap='coolwarm' )
plt.show()
```



[32]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

[33]:
```
df['Species'] = le.fit_transform(df['Species'])
df.head(150)
```

[33]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	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
..             …             …              …             …          …
145            6.7           3.0            5.2           2.3        2
146            6.3           2.5            5.0           1.9        2
147            6.5           3.0            5.2           2.0        2
148            6.2           3.4            5.4           2.3        2
149            5.9           3.0            5.1           1.8        2

[150 rows x 5 columns]
```

[43]:
```
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)
```

[44]:
```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

[45]:
```
# model trainin
model.fit(x_train,y_train)
```

```
C:\Users\Admin\anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:814: 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(
```

[45]:
```
LogisticRegression()
```

[46]:
```
#metrix
print("Accuracy: ",model.score(x_test, y_test)*100)
```

```
Accuracy:  95.55555555555556
```

[47]:
```
print("Recall: ",model.score(x_test, y_test)*100)
```

```
Recall:  95.55555555555556
```

[48]:
```
print("Precision: ",model.score(x_test, y_test)*100)
```

```
Precision:  95.55555555555556
```

[ ]:
```

```