Iris Species Classification

Import dataset

In [20]:
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as  plt
import seaborn as sns
```

In [21]:
```
df= pd.read_csv('Iris.csv')
df.head(10)
```

Out[21]:

	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
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

In [22]:
```
# delete a column
df= df.drop(columns=['Id'])
df.head()
```

Out[22]:

	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

In [23]:
```
df.shape
```

Out[23]:
```
(150, 5)
```

In [24]:
```
#To display statistics about data
df.describe()
```

Out[24]:

	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

In [25]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [26]:
```
df.sample(5)
```

Out[26]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
75	6.6	3.0	4.4	1.4	Iris-versicolor
32	5.2	4.1	1.5	0.1	Iris-setosa
73	6.1	2.8	4.7	1.2	Iris-versicolor
6	4.6	3.4	1.4	0.3	Iris-setosa
96	5.7	2.9	4.2	1.3	Iris-versicolor

Preprocessing the dataset

In [27]:
```
#Check for null values
df.isnull().sum()
```

Out[27]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [28]:
```
df.duplicated().sum()
```

Out[28]:
```
3
```

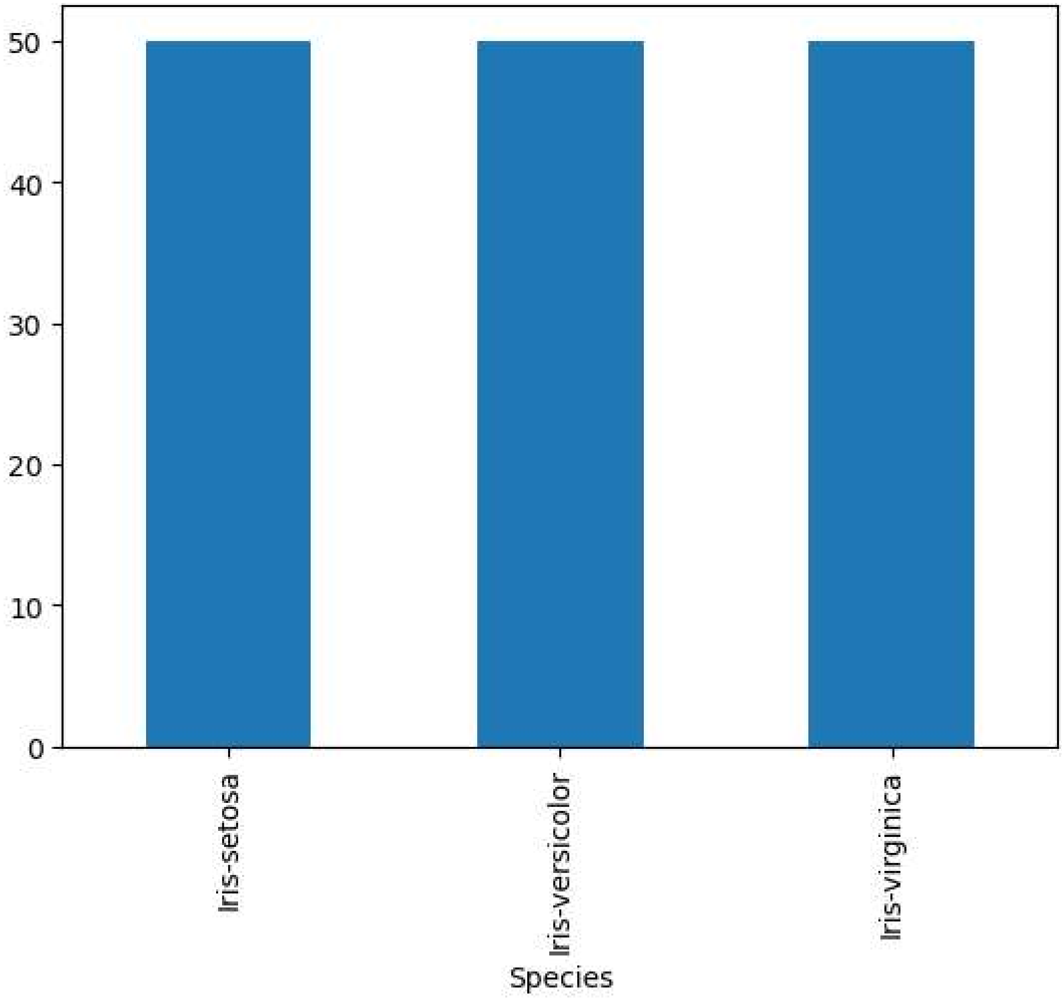
In [29]:
```
#To display no of sample on each class
df['Species'].value_counts()
```

Out[29]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

Exploratory Data Analysis

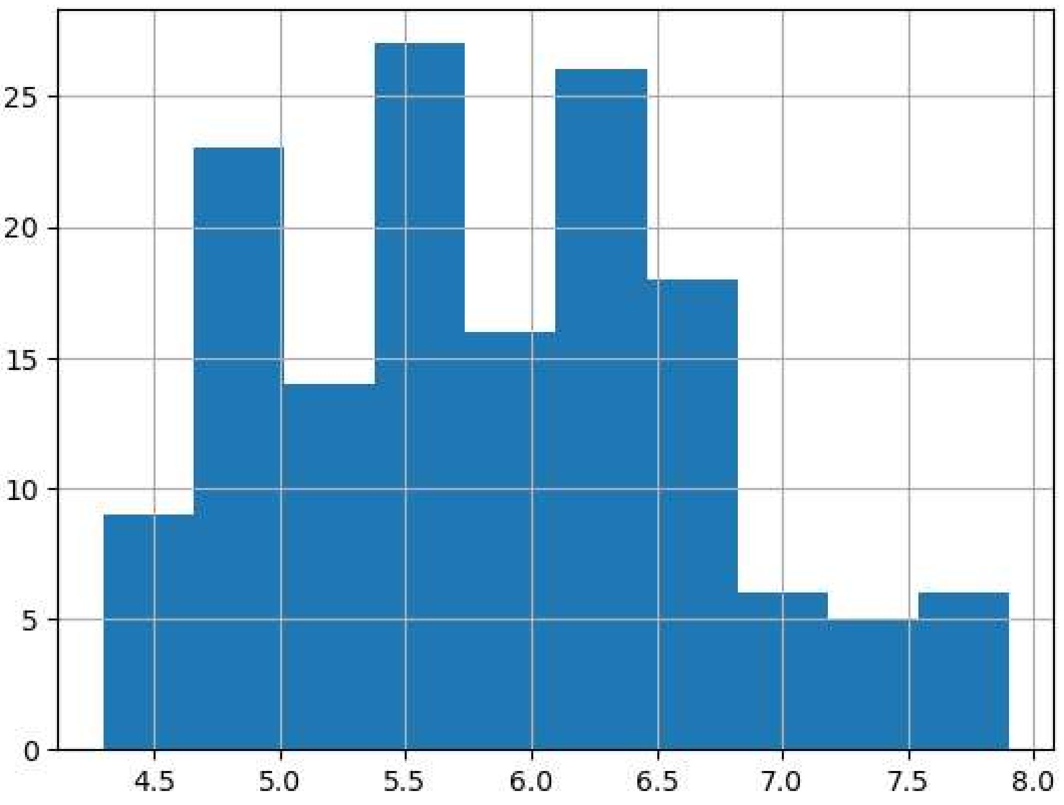
In [30]:
```
df['Species'].value_counts().plot(kind='bar')
```

Out[30]:
```
<Axes: xlabel='Species'>
```



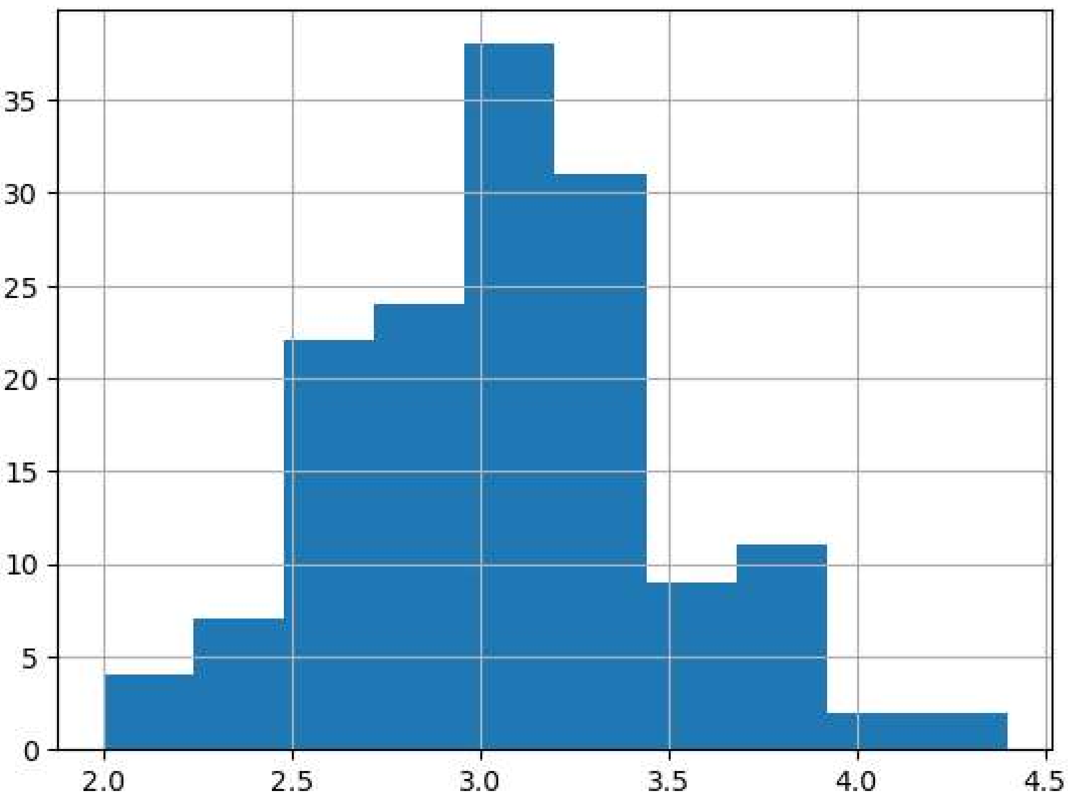
In [31]:
```
df['SepalLengthCm'].hist()
```

Out[31]: <Axes: >



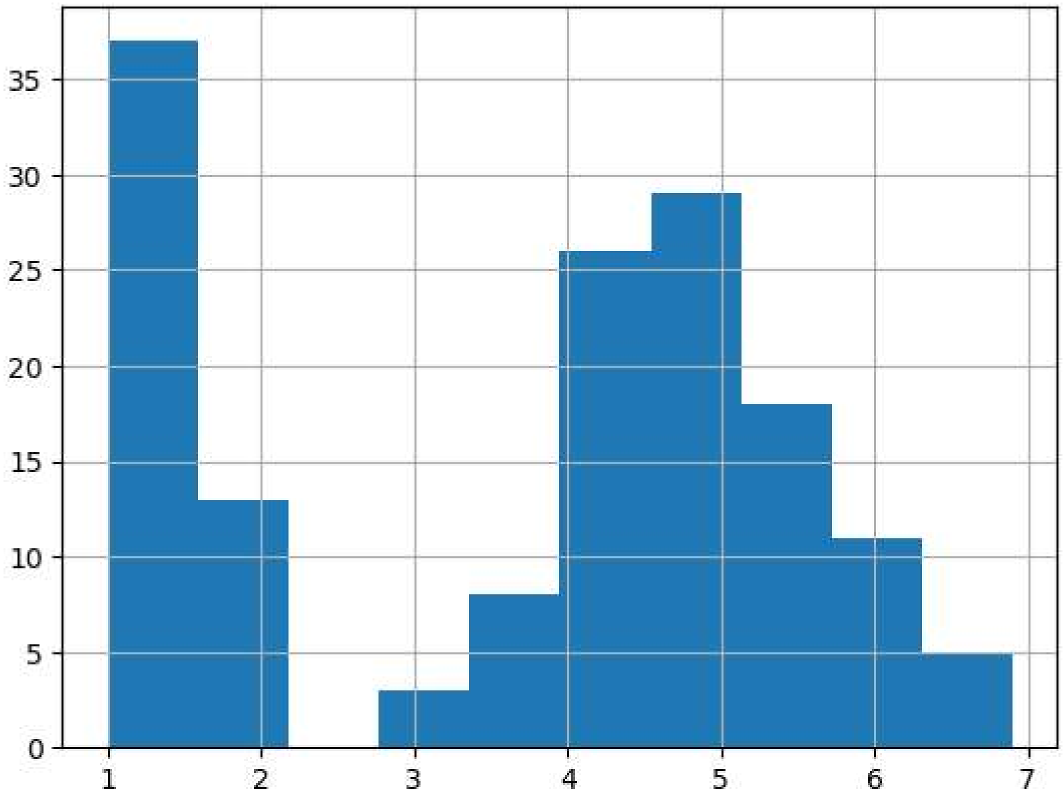
In [32]:
```
df['SepalWidthCm'].hist()
```

Out[32]: <Axes: >



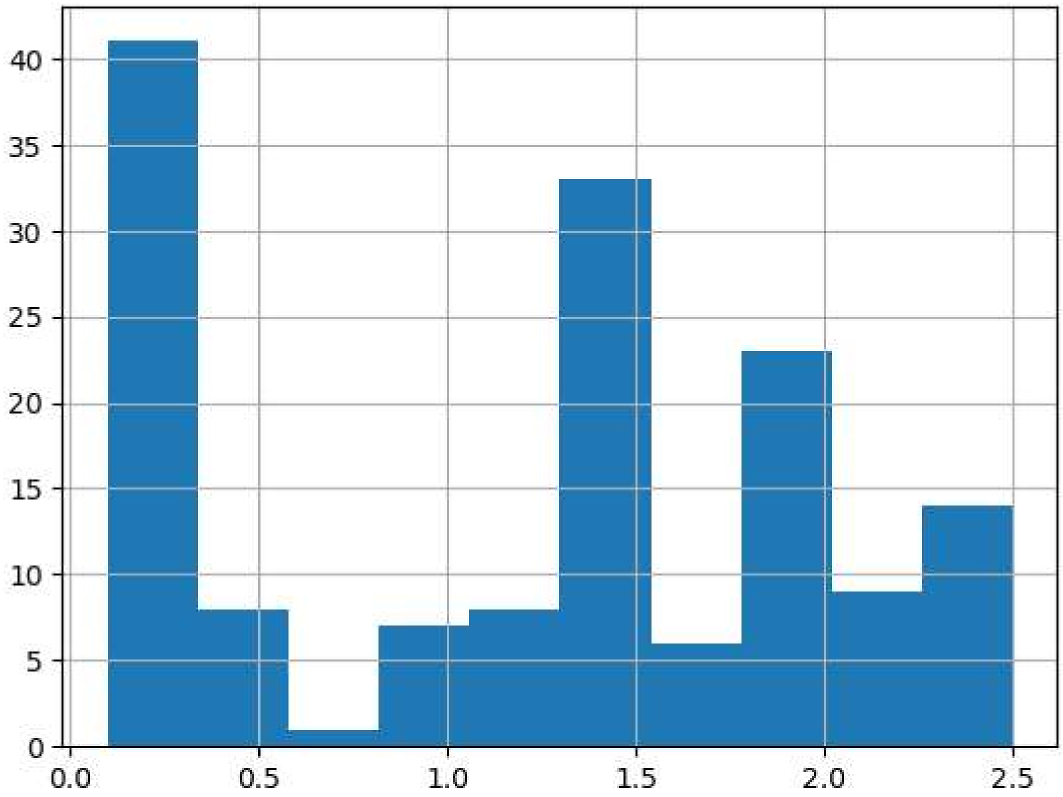
In [33]:
```
df['PetalLengthCm'].hist()
```

Out[33]: <Axes: >



In [34]:
```
df['PetalWidthCm'].hist()
```

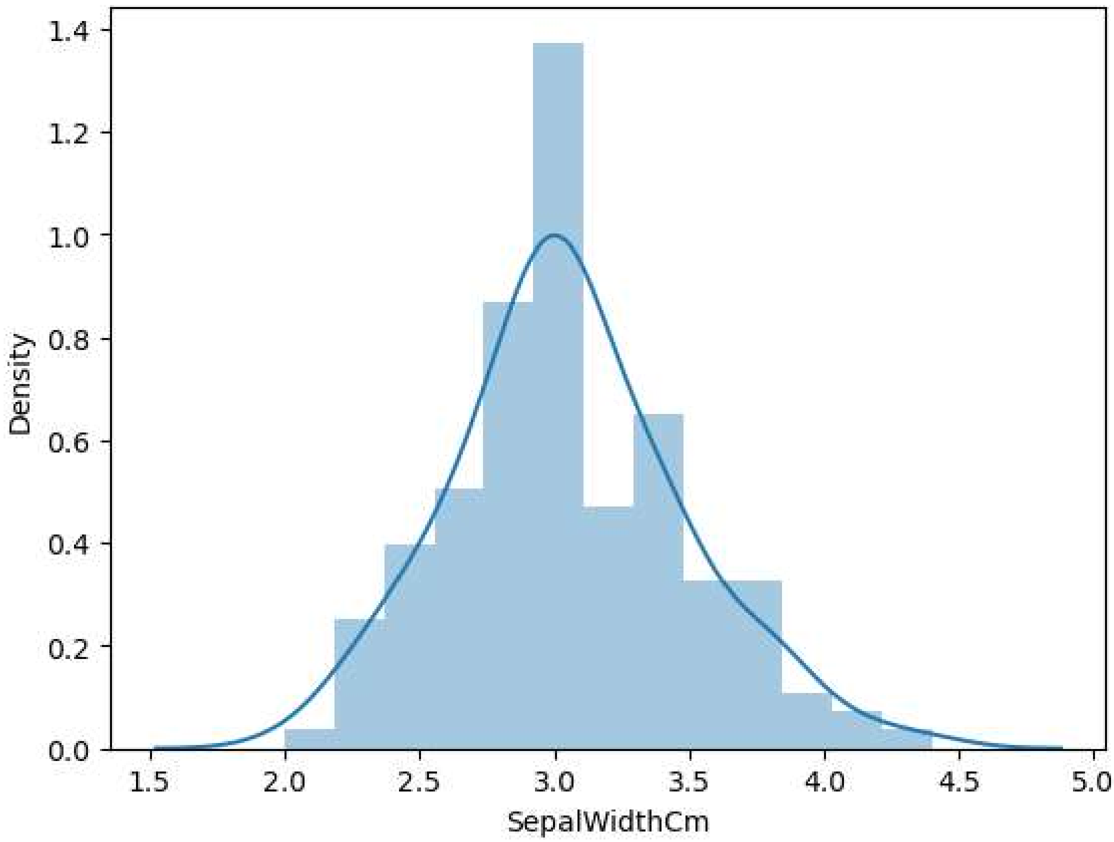
Out[34]: <Axes: >



In [35]:
```
sns.distplot(df['SepalWidthCm'])
```

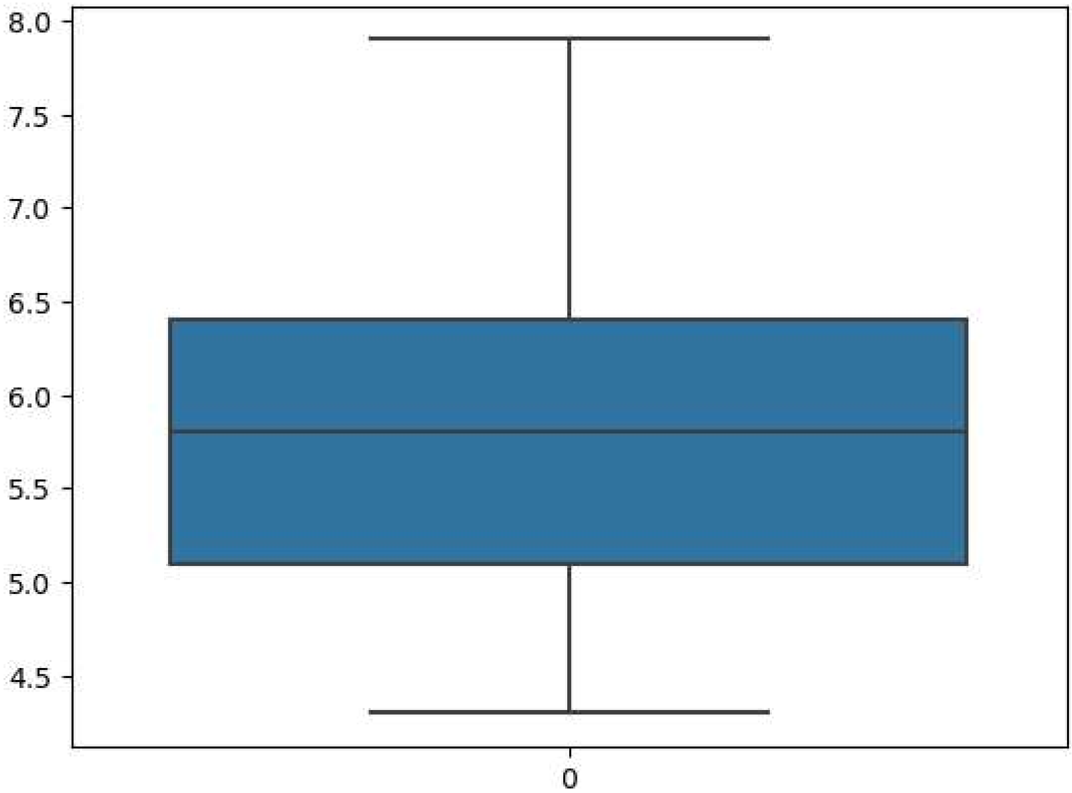
```
C:\Users\DELL\AppData\Local\Temp\ipykernel_12948\3402425195.py:1: UserWarn
ing:

`distplot` is a deprecated function and will be removed in seaborn v0.14.
0.

Please adapt your code to use either `displot` (a figure-level function wi
th
similar flexibility) or `histplot` (an axes-level function for histogram
s).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://
gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(df['SepalWidthCm'])
```

Out[35]:
```
<Axes: xlabel='SepalWidthCm', ylabel='Density'>
```



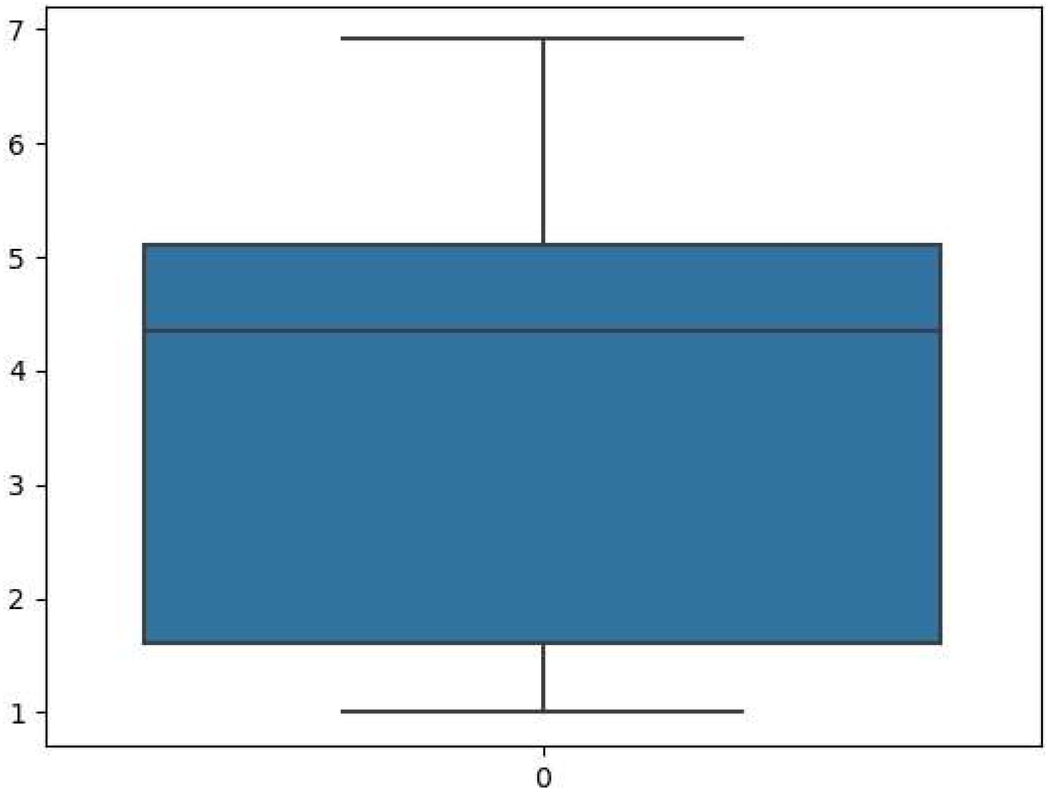
In [36]:
```
sns.boxplot(df['SepalLengthCm'])
```

Out[36]: <Axes: >



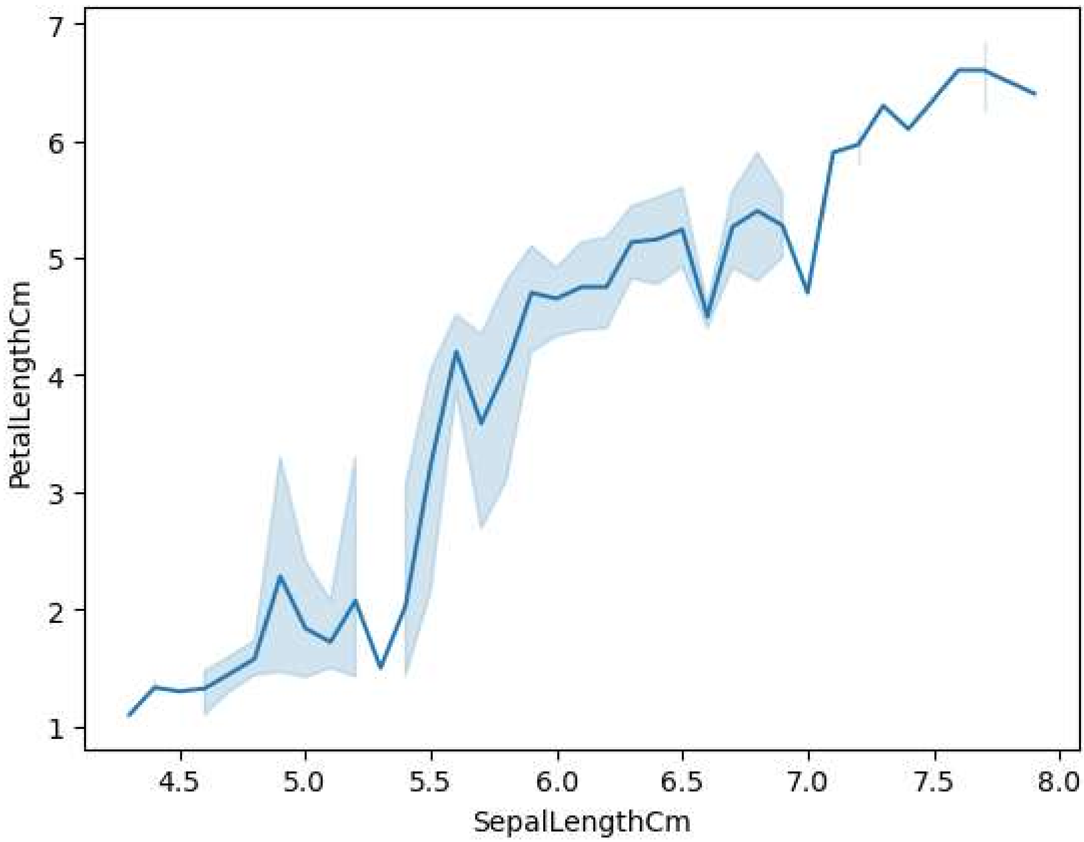
In [37]:
```
sns.boxplot(df['PetalLengthCm'])
```

Out[37]: <Axes: >

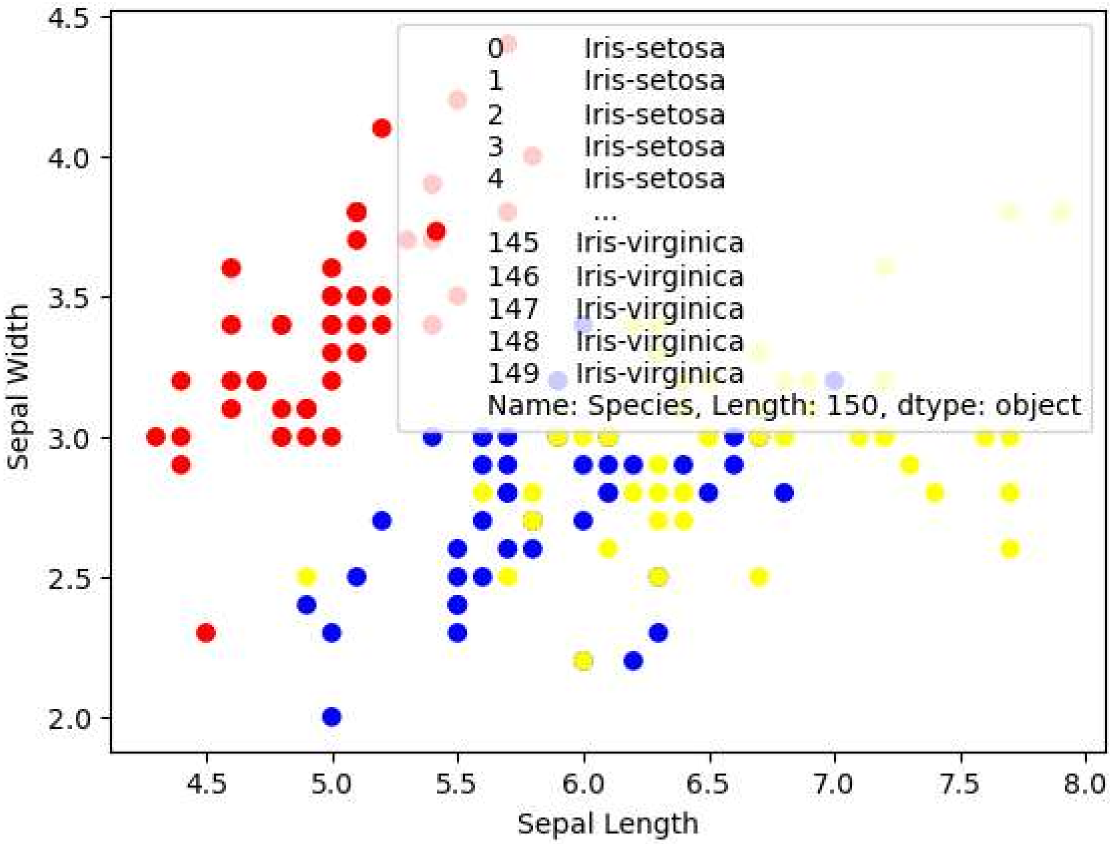


In [64]:
```
sns.lineplot(data=df, x='SepalLengthCm', y='PetalLengthCm')
```

Out[64]: <Axes: xlabel='SepalLengthCm', ylabel='PetalLengthCm'>

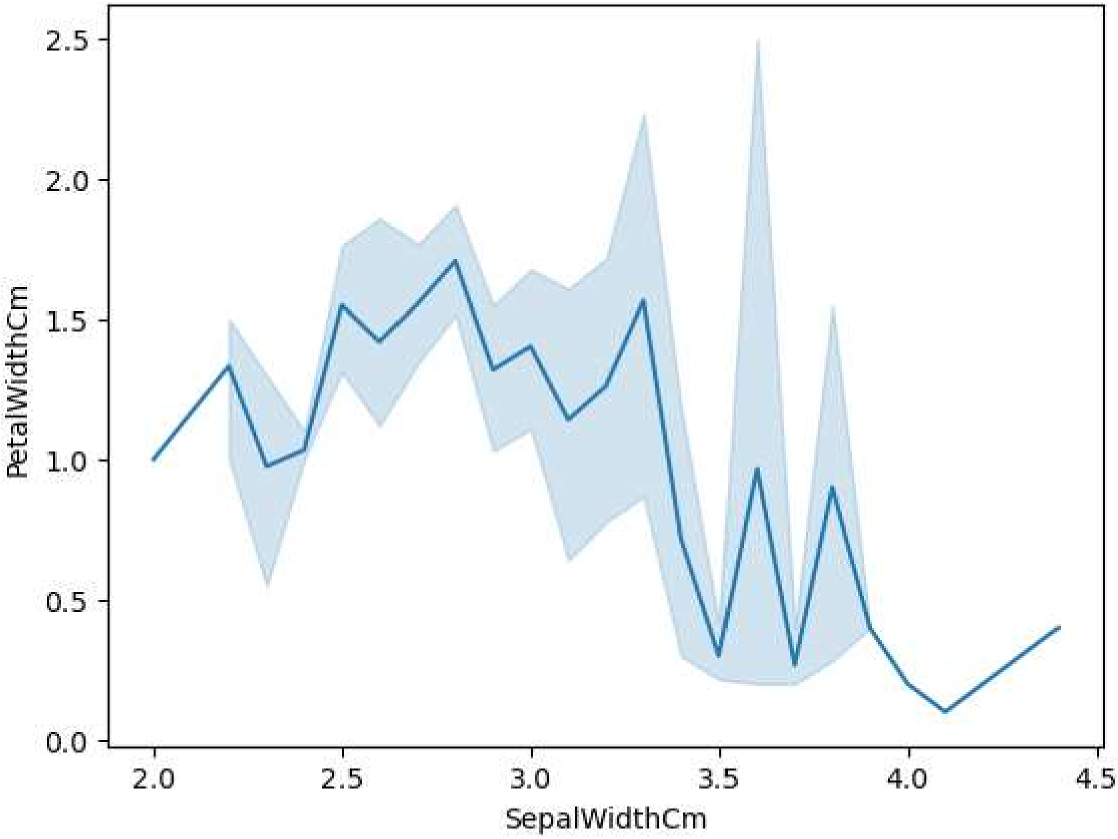


In [38]:
```
#scatterplot
colors = {'Iris-setosa': 'red', 'Iris-versicolor': 'blue', 'Iris-virginica':

# Create a scatterplot
plt.scatter(df['SepalLengthCm'], df['SepalWidthCm'], c=[colors[species] for

# Add labels
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width ')
plt.legend()

# Show the plot
plt.show()
```



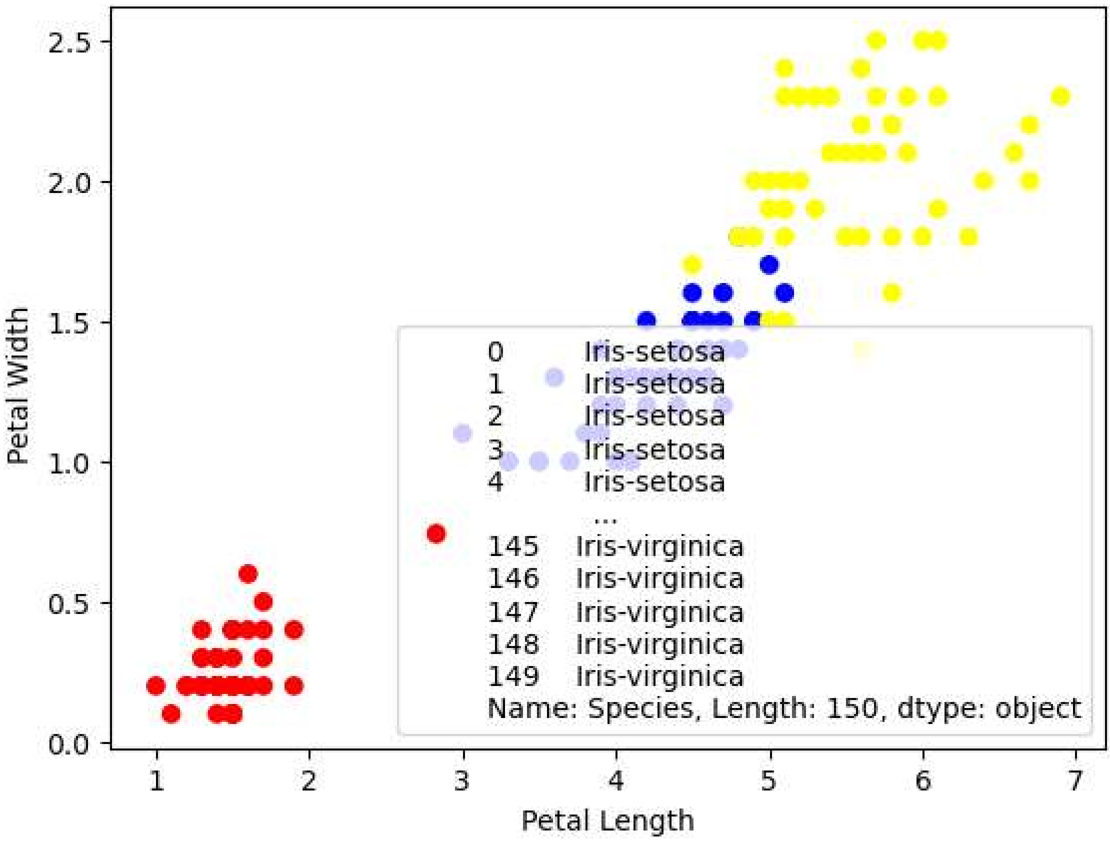
In [65]:
```
sns.lineplot(data=df, x='SepalWidthCm', y='PetalWidthCm')
```

Out[65]: <Axes: xlabel='SepalWidthCm', ylabel='PetalWidthCm'>

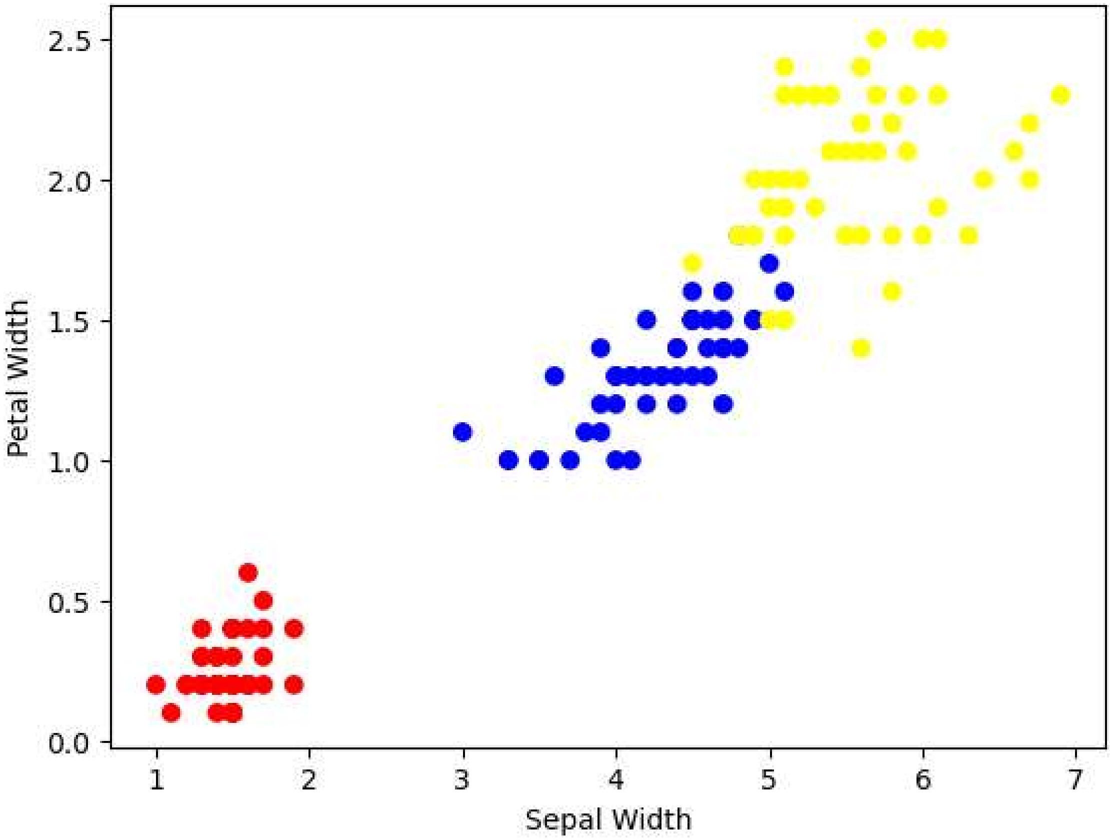


In [39]:
```
colors = {'Iris-setosa': 'red', 'Iris-versicolor': 'blue', 'Iris-virginica':

# Create a scatterplot
plt.scatter(df['PetalLengthCm'], df['PetalWidthCm'], c=[colors[species] for

# Add labels
plt.xlabel('Petal Length')
plt.ylabel('Petal Width ')
plt.legend()
```

Out[39]: <matplotlib.legend.Legend at 0x22e4d625f50>



In [40]:
```
plt.scatter(df['PetalLengthCm'], df['PetalWidthCm'], c=[colors[species] for
plt.xlabel("Sepal Width")
plt.ylabel("Petal Width")

plt.show()
```



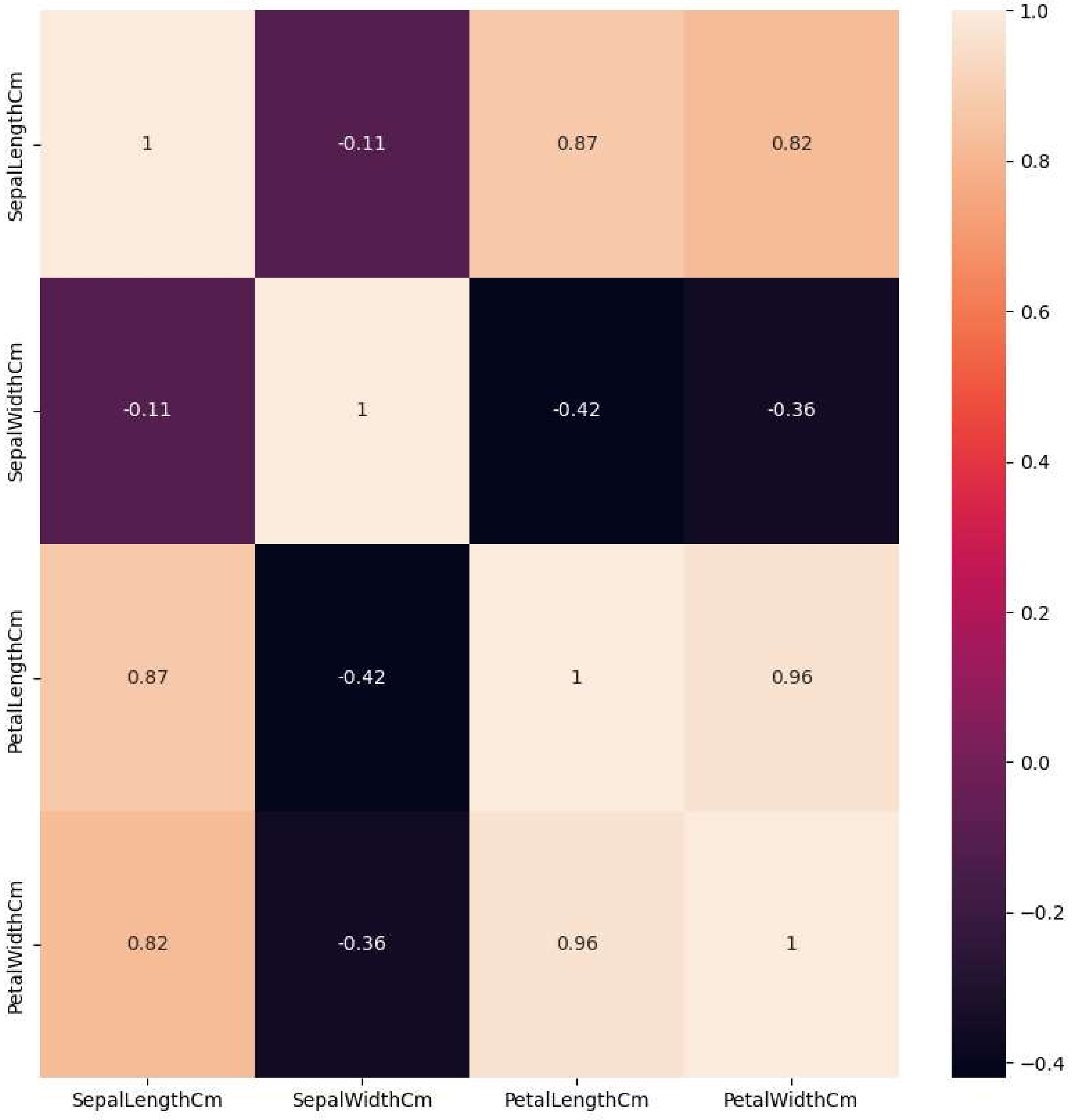
Coorelation Matrix

A correlation matrix is a table showing correlation coefficients between variable .Each cell in the table shows the correlation between two variables. The value is in the range of -1 to 1. If two varaible have high correlation,we can neglect one variable from those two.

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

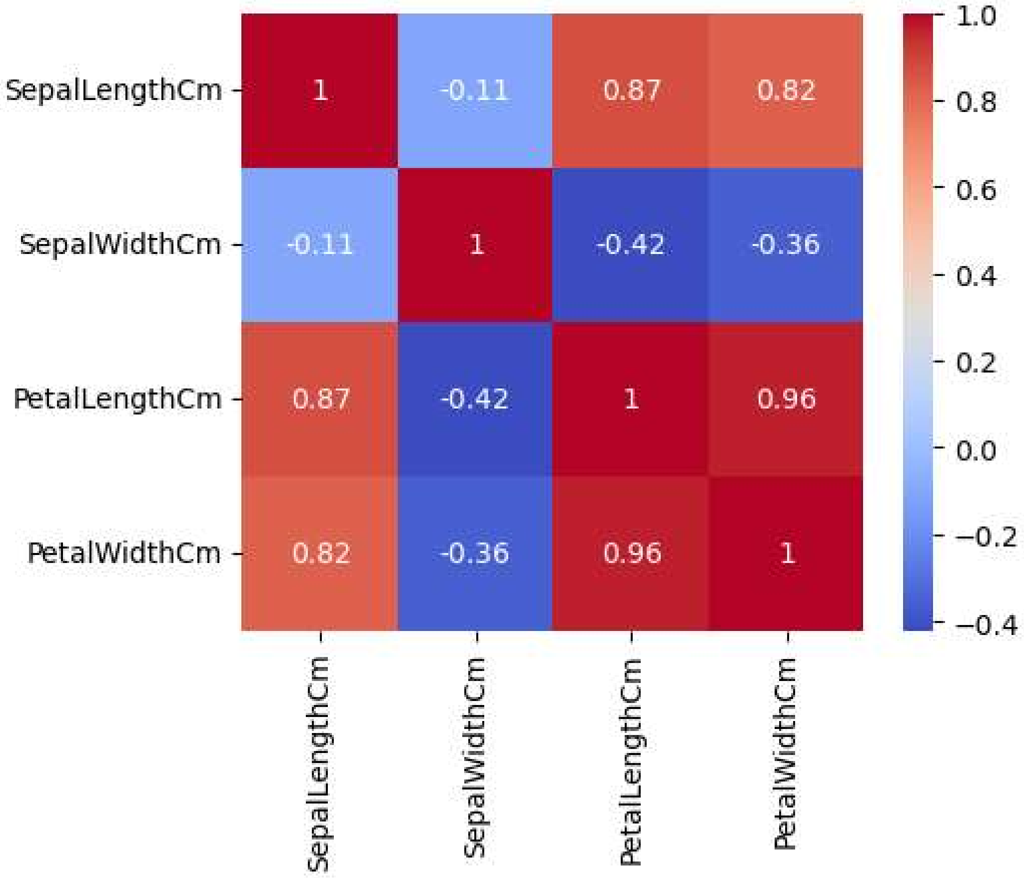
In [42]:
```
corr = df.select_dtypes(include=['float64']).corr()
fig, ax = plt.subplots(figsize=(10, 10))
sns.heatmap(corr, annot=True, ax=ax)
```

Out[42]: <Axes: >



In [43]:
```
corr = df.select_dtypes(include=['float64']).corr()
fig, ax = plt.subplots(figsize=(5,4))
sns.heatmap(corr, annot=True, ax=ax,cmap='coolwarm')
```

Out[43]: <Axes: >



Label Encoder

In [44]:
```
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
```

In [45]:
```
df['Species']=le.fit_transform(df['Species'])
df.head()
```

Out[45]:

	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

Model Training

In [46]:
```
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)
```

In [47]:
```
model=LogisticRegression()
```

In [48]:
```
model.fit(x_train,y_train)
```

Out[48]:
```
▸ LogisticRegression
```

In [49]:
```
#print metric to get performance
print("Accuracy:",model.score(x_test,y_test)*100)
```

```
Accuracy: 97.77777777777777
```

In [50]:
```
#knn- k-nearest neighbours
from sklearn.neighbors import KNeighborsClassifier
```

In [51]:
```
model.fit(x_train,y_train)
```

Out[51]:
```
▸ LogisticRegression
```

In [52]:
```
#print metric to get performance
print("Accuracy:",model.score(x_test,y_test)*100)
```

```
Accuracy: 97.77777777777777
```

In [53]:
```
#decision tree
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier()
model.fit(x_train,y_train)
```

Out[53]:
```
▸ DecisionTreeClassifier
```

In [57]:
```
#Random forest Regression
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(x_train,y_train)
print("Accuracy:",model.score(x_test,y_test)*100)
```

```
Accuracy: 93.33333333333333
```

In [63]:
```
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.model_selection import cross_val_score
model = ExtraTreesClassifier()
model.fit(x_train, y_train)
print('Accuracy:', model.score(x_test, y_test))

# Now define and use X and y for cross-validation
score = cross_val_score(model, x_train,y_train, cv=5)
print('CV Score:', np.mean(score))
```

```
Accuracy: 0.9555555555555556
CV Score: 0.9523809523809523
```

In [ ]: