

```
In [1]: import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
In [2]: df = pd.read_csv('Iris.csv')
df.head()
```

```
Out[2]:
```

	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

```
In [3]: df = df.drop(columns=["Id"])
df.head()
```

```
Out[3]:
```

	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 [4]: df.head(10)
```

```
Out[4]:
```

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

```
In [5]: df.describe()
```

```
Out[5]:
```

	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 [6]: 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 [7]: df['Species'].value_counts()
```

```
Out[7]:
```

```
Species
Iris-setosa    50
Iris-versicolor    50
Iris-virginica    50
Name: count, dtype: int64
```

```
In [8]: from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Species'] = le.fit_transform(df['Species'])
df['Species']
```

```
Out[8]:
```

```
0    0
1    0
2    0
3    0
4    0
..
145  2
146  2
147  2
148  2
149  2
Name: Species, Length: 150, dtype: int32
```

```
In [9]: df.dtypes
```

```
Out[9]:
```

```
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species          int32
dtype: object
```

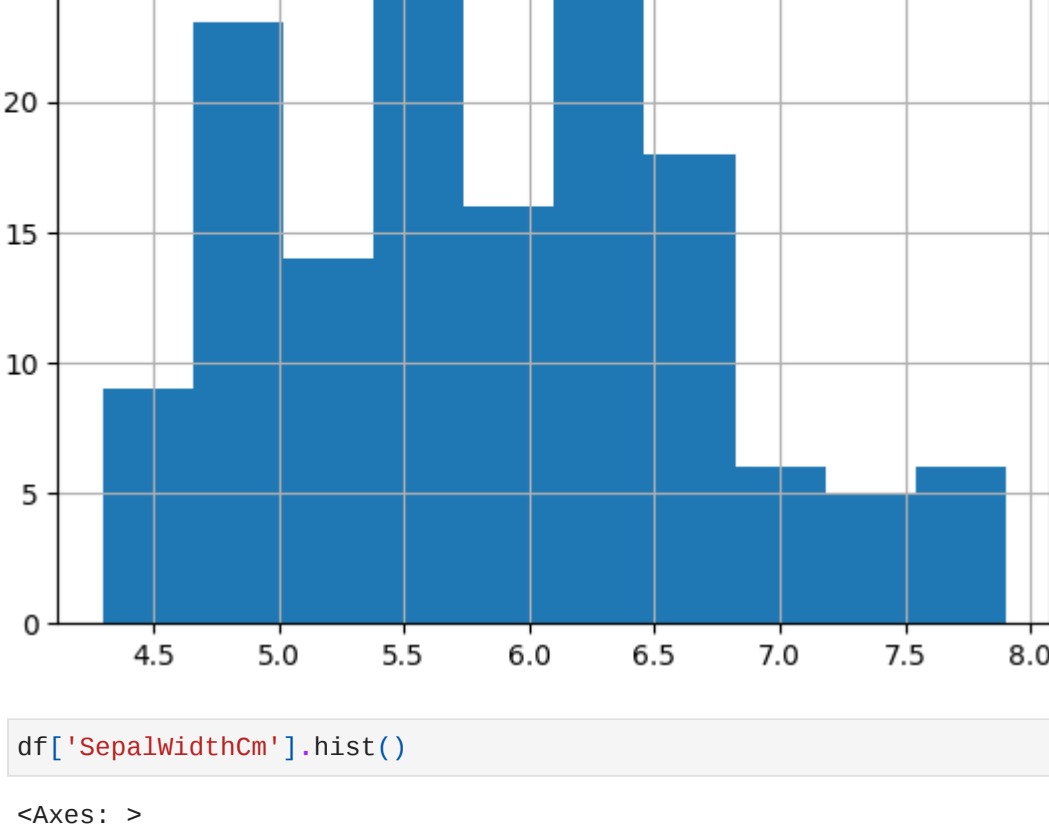
```
In [10]: df.isnull().sum()
```

```
Out[10]:
```

```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

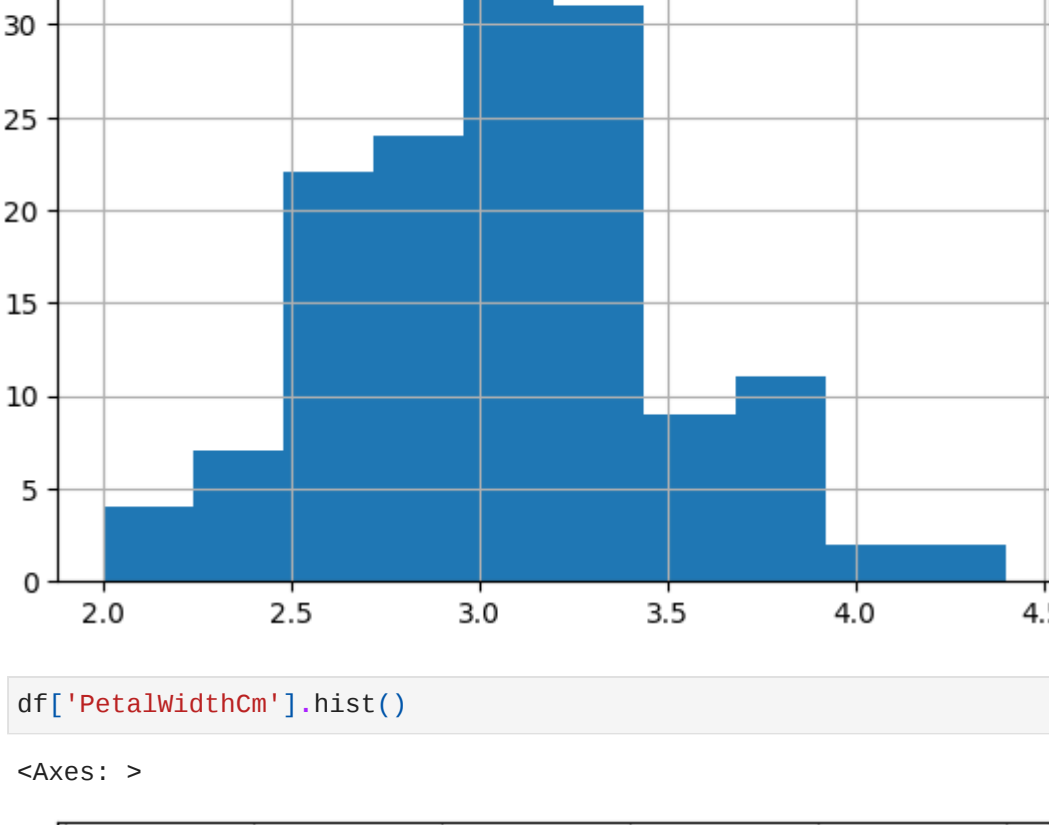
```
In [11]: df['SepalLengthCm'].hist()
```

```
Out[11]:
```



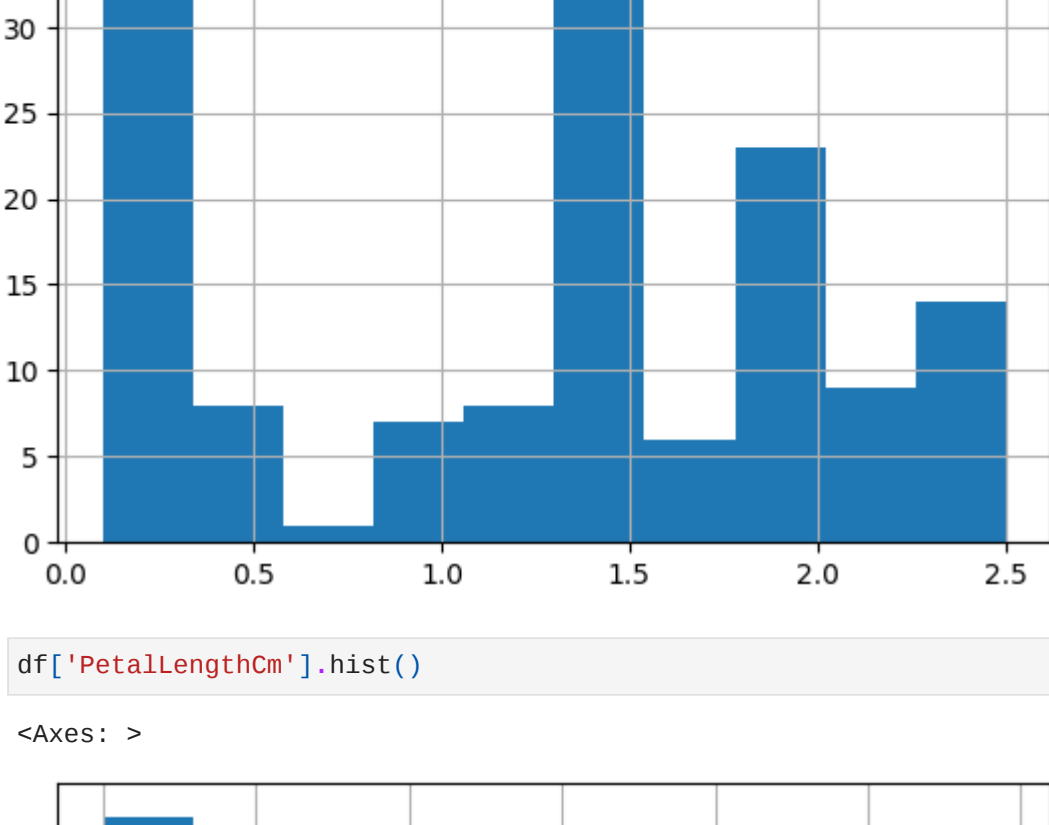
```
In [12]: df['SepalWidthCm'].hist()
```

```
Out[12]:
```



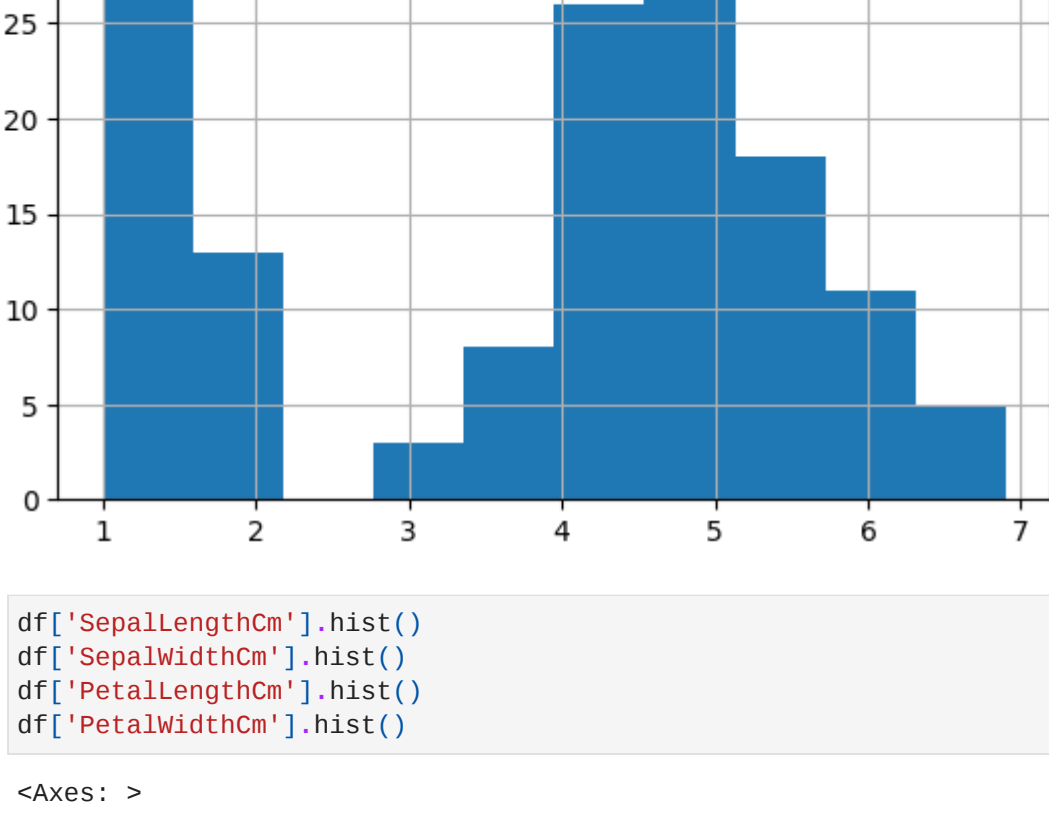
```
In [13]: df['PetalWidthCm'].hist()
```

```
Out[13]:
```



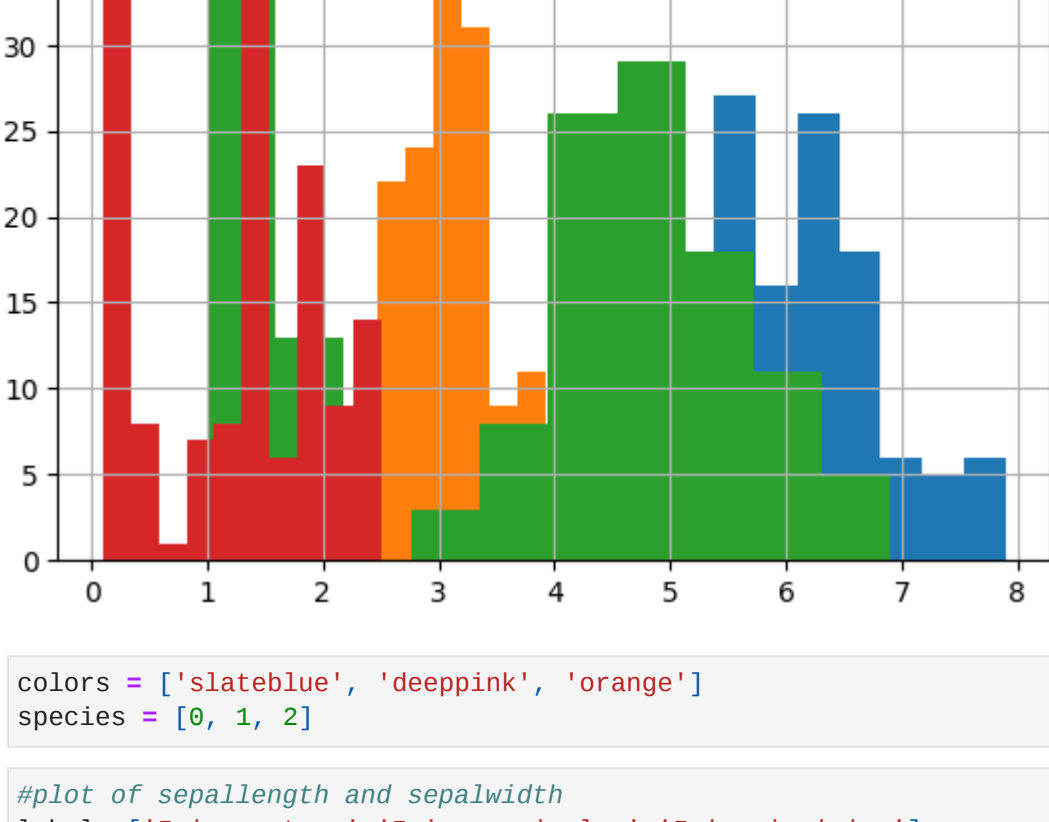
```
In [14]: df['PetalLengthCm'].hist()
```

```
Out[14]:
```



```
In [15]: df['SepalLengthCm'].hist()
df['SepalWidthCm'].hist()
df['PetalLengthCm'].hist()
df['PetalWidthCm'].hist()
```

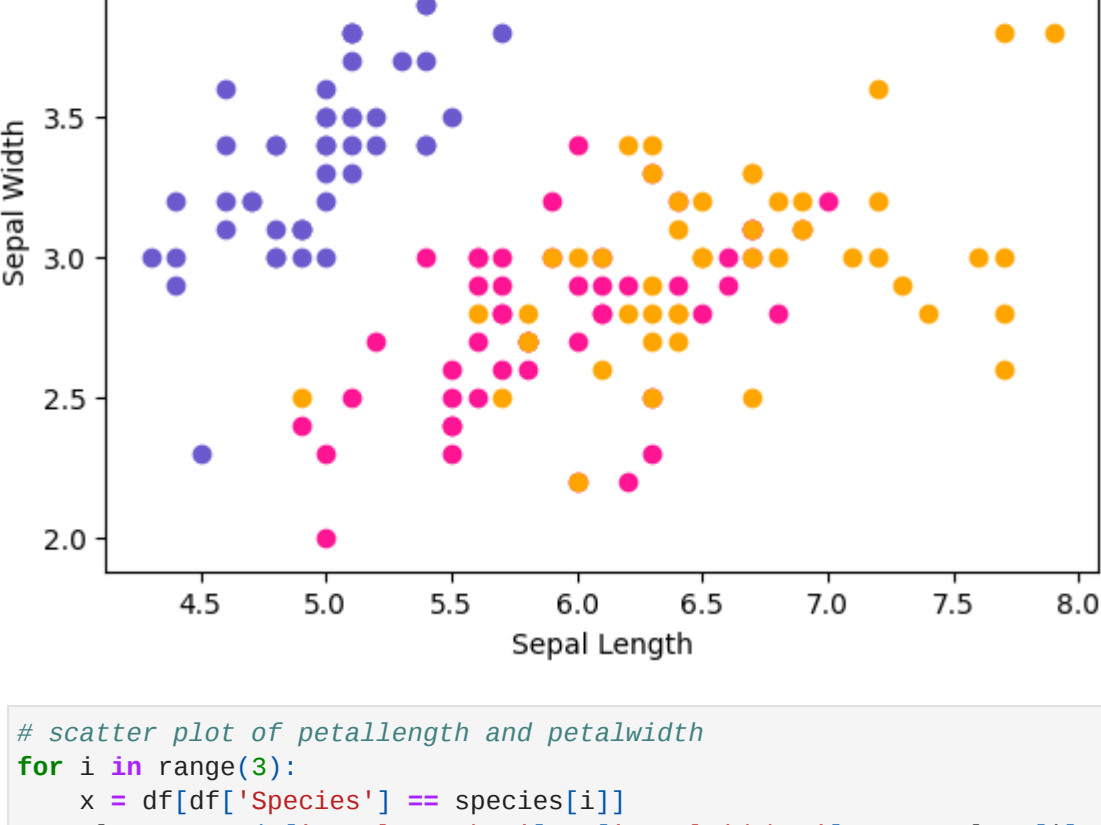
```
Out[15]:
```



```
In [16]: colors = ['slateblue', 'deeppink', 'orange']
species = [0, 1, 2]
```

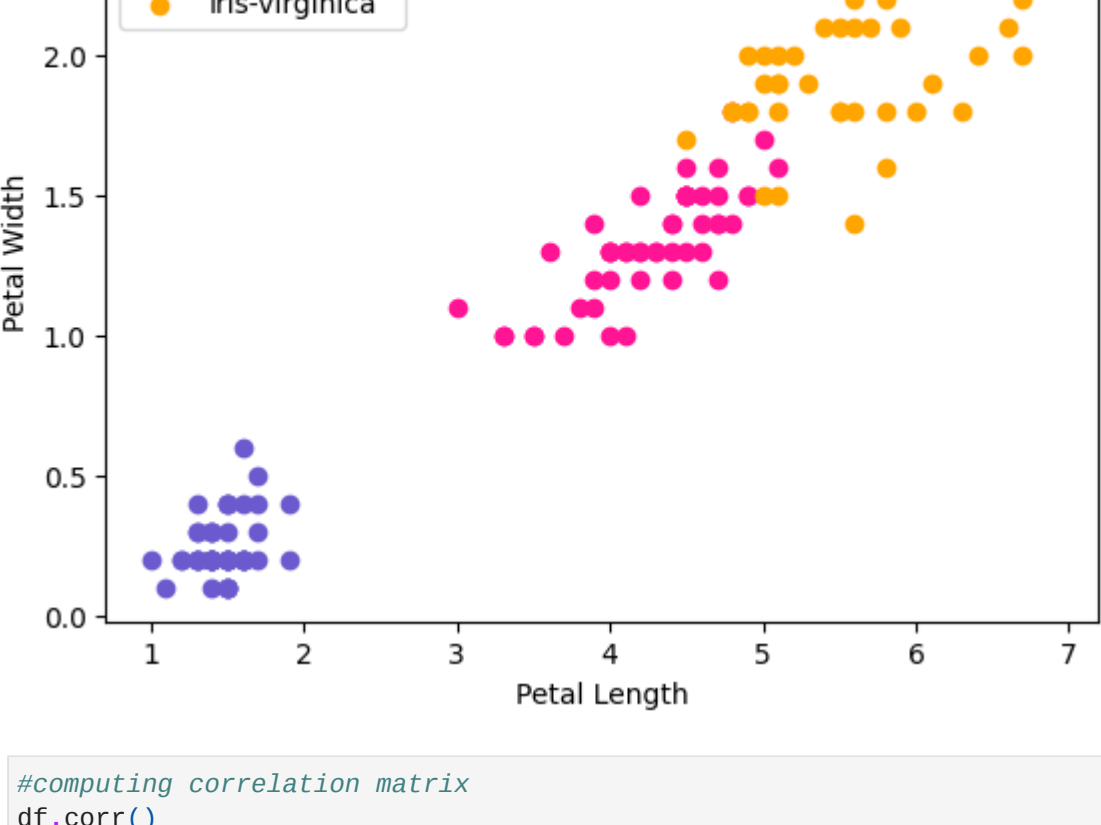
```
In [17]: #plot of sepal length and sepal width
labels=['Iris-setosa','Iris-versicolor','Iris-virginica']
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['SepalLengthCm'], x['SepalWidthCm'], c=colors[i], label=labels[i])
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.legend()
```

```
Out[17]:
```



```
In [18]: # scatter plot of petal length and petal width
for i in range(3):
    x = df[df['Species'] == species[i]]
    plt.scatter(x['PetalLengthCm'], x['PetalWidthCm'], c = colors[i], label=labels[i])
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
plt.legend()
```

```
Out[18]:
```



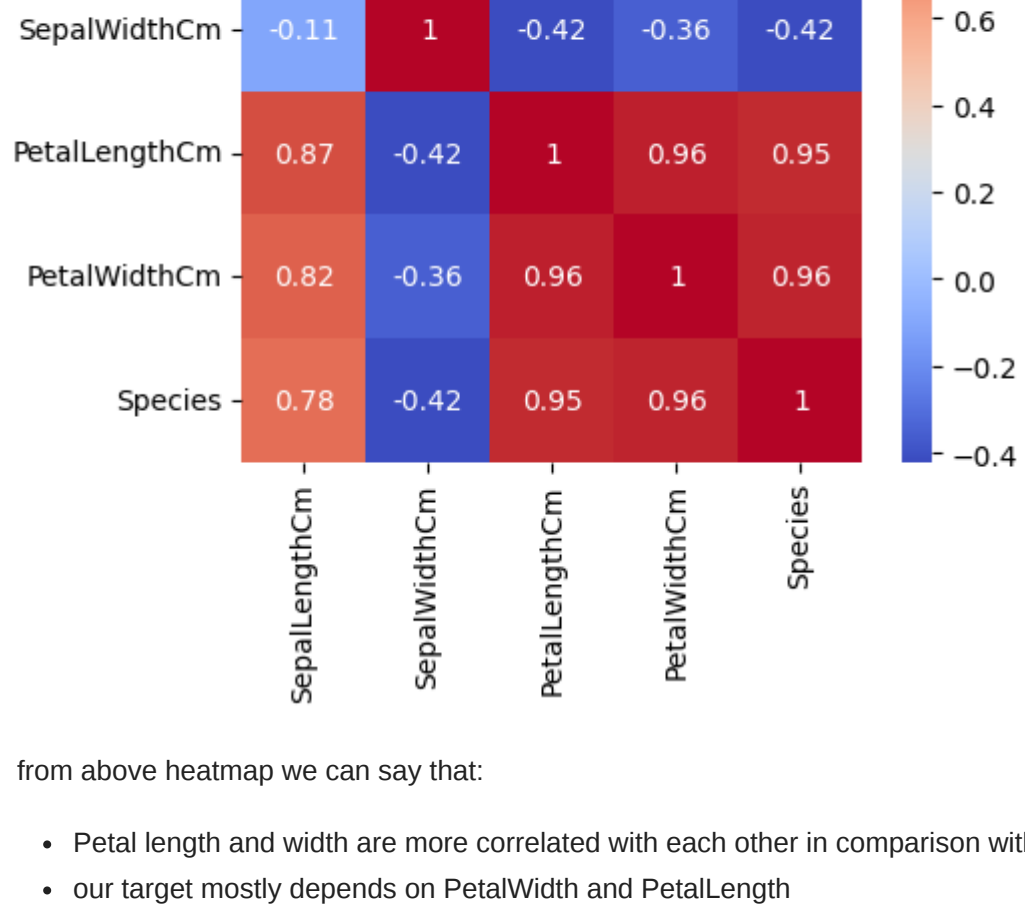
```
In [19]: #computing correlation matrix
df.corr()
```

```
Out[19]:
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954	0.782561
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544	-0.419446
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757	0.949043
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000	0.956464
Species	0.782561	-0.419446	0.949043	0.956464	1.000000

```
In [20]: #A heat map is a 2-dimensional data visualization technique that represents the magnitude of individual values within a dataset as a color.
corr = df.corr()
fig, ax = plt.subplots(figsize=(5, 4))
sns.heatmap(corr, annot=True, ax=ax, cmap='coolwarm')
#annot used for annotating in the graph
#correlation is used for comparing relation among datasets
```

```
Out[20]:
```



from above heatmap we can say that:

- Petal length and width are more correlated with each other in comparison with Sepal length and width
- our target mostly depends on PetalWidth and PetalLength
- Sepal width has negative correlation so we can drop it to make our results more accurate and apply data cleaning

```
In [34]: from sklearn.model_selection import train_test_split
X = df.drop(columns=['Species', 'SepalWidthCm'])
Y = df['Species']
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.40)
```

```
In [35]: from sklearn.linear_model import LogisticRegression
model1 = LogisticRegression()
model1.fit(X_train, y_train)
print("Accuracy (Logistic Regression): ", model1.score(X_test, y_test) * 100)
```

```
Accuracy (Logistic Regression):  96.66666666666667
```

```
In [36]: # K-nearest Neighbours Model (KNN)
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier()
model2.fit(X_train, y_train)
print("Accuracy (KNN): ", model2.score(X_test, y_test) * 100)
```

```
Accuracy (KNN):  98.33333333333333
```

```
In [37]: # Decision Tree Model
from sklearn.tree import DecisionTreeClassifier
model3 = DecisionTreeClassifier()
```

```
model3.fit(x_train, y_train)
print("Accuracy (Decision Tree): ", model3.score(x_test, y_test) * 100)
```

```
Accuracy (Decision Tree): 96.66666666666667
```

In []: