DATASET INFORMATION

The Dataset contains 3 classes of 50 instances each...where each class refers to a type of iris plant. One class is linearly separable from the other 2, The latter are not linearly separable from each other

Attributes Information:

```
1. sepal_length in cm
2. sepal_width in cm
3. petal_length in cm
4. petal_width in cm
5. class -- Iris Setosa - Iris Versicolour - iris Vigrinica
6. Number of Instances: 150 (50 in each of three classes)
7. Number of Attributes: 4 numeric,1 object
8. Class Distribution: 33.3% for each of 3 classes
```

IMPORT MODULES

In [1]:
```
#importing libraries for further use
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
import sklearn.preprocessing
import scipy
```

LOAD DATASET

In [2]:
```
#loading the data
df = pd.read_csv('IRIS.csv')
```

In [3]:
```
#vieweing the dataset for top 5 rows
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.shape
```

Out[4]:
```
(150, 5)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

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

Out[6]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [7]:
```
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
```

SUMMARIZATION OF DATA

In [8]:
```
df.describe()
```

Out[8]:

	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

PREPROCESSING THE DATASET

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

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

In [10]:
```
#Showing nullvalues in percentage format
missing_values = df.isnull().sum()
total = df.isnull().sum().sort_values(ascending = False)
percent = ((df.isnull().sum()/df.shape[0]*100))
percent = percent.sort_values(ascending = False)
missing_data = pd.concat([total,percent],axis = 1,
                        keys = ['Total Missing Values','Percentage of Missing
missing_data['data(dtypes)'] = df[missing_data.index].dtypes
missing_data
```

Out[10]:

	Total Missing Values	Percentage of Missing values	data(dtypes)
sepal_length	0	0.0	float64
sepal_width	0	0.0	float64
petal_length	0	0.0	float64
petal_width	0	0.0	float64
species	0	0.0	object

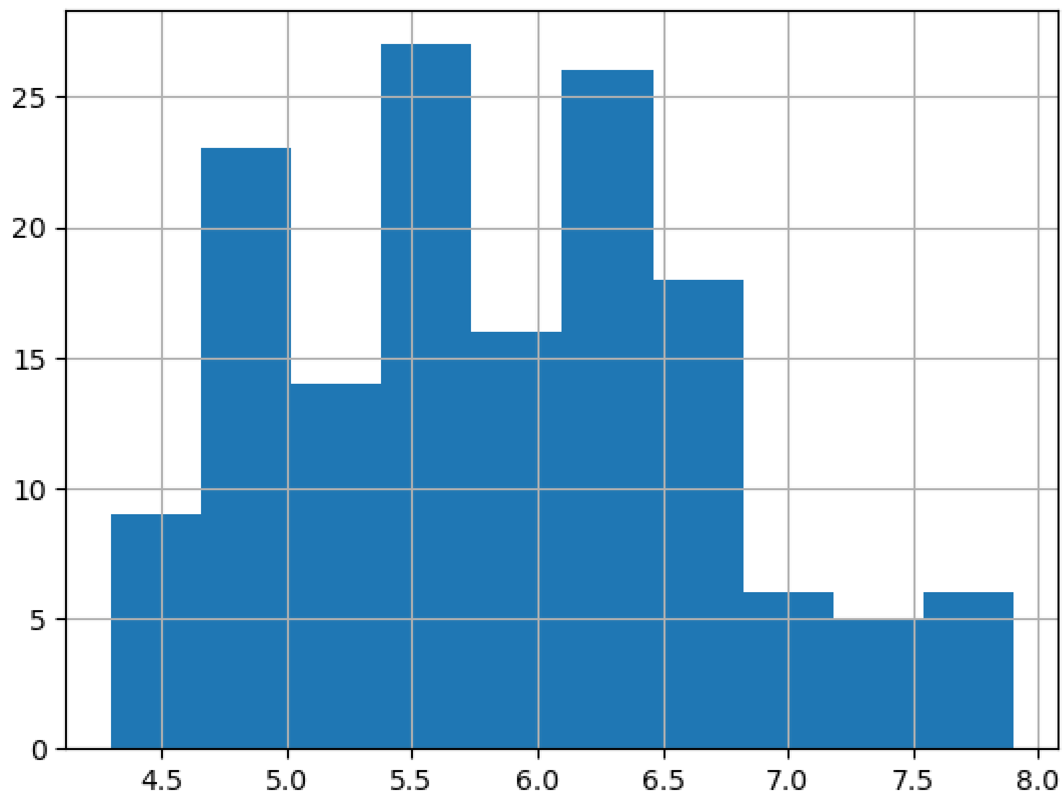
In [11]:
```
#visualization of NaN values using heatmap
sns.heatmap(df.isnull(),cbar = False)
```

Out[11]: <AxesSubplot:>

EXPLORATORY DATA ANALYSIS

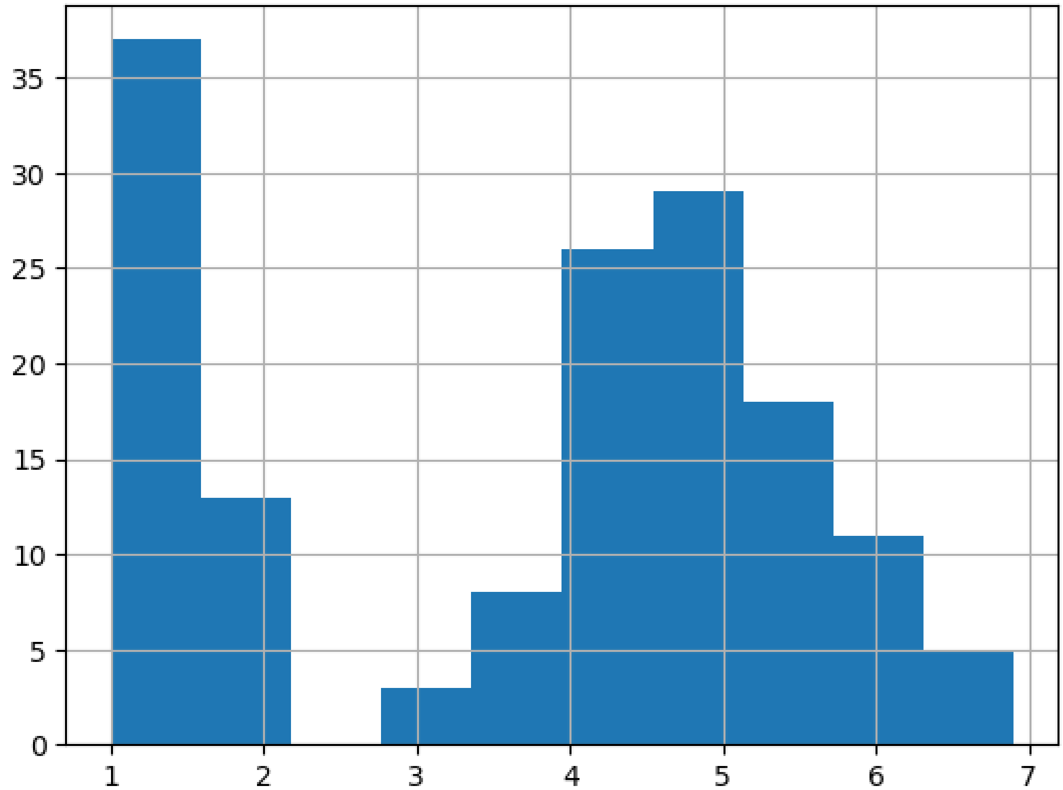
In [12]:
```
#Analyzing using histogram for columns in dataset
df['sepal_length'].hist()
```

Out[12]: <AxesSubplot:>



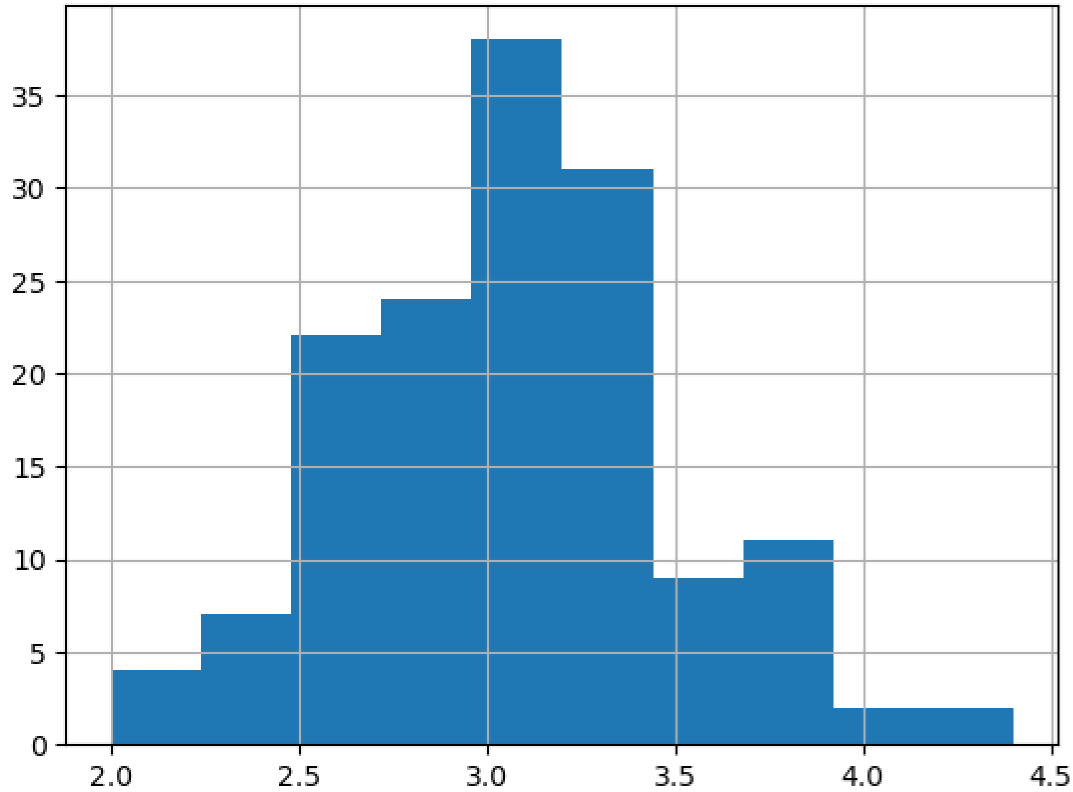
In [13]:
```
df['petal_length'].hist()
```

Out[13]: <AxesSubplot:>



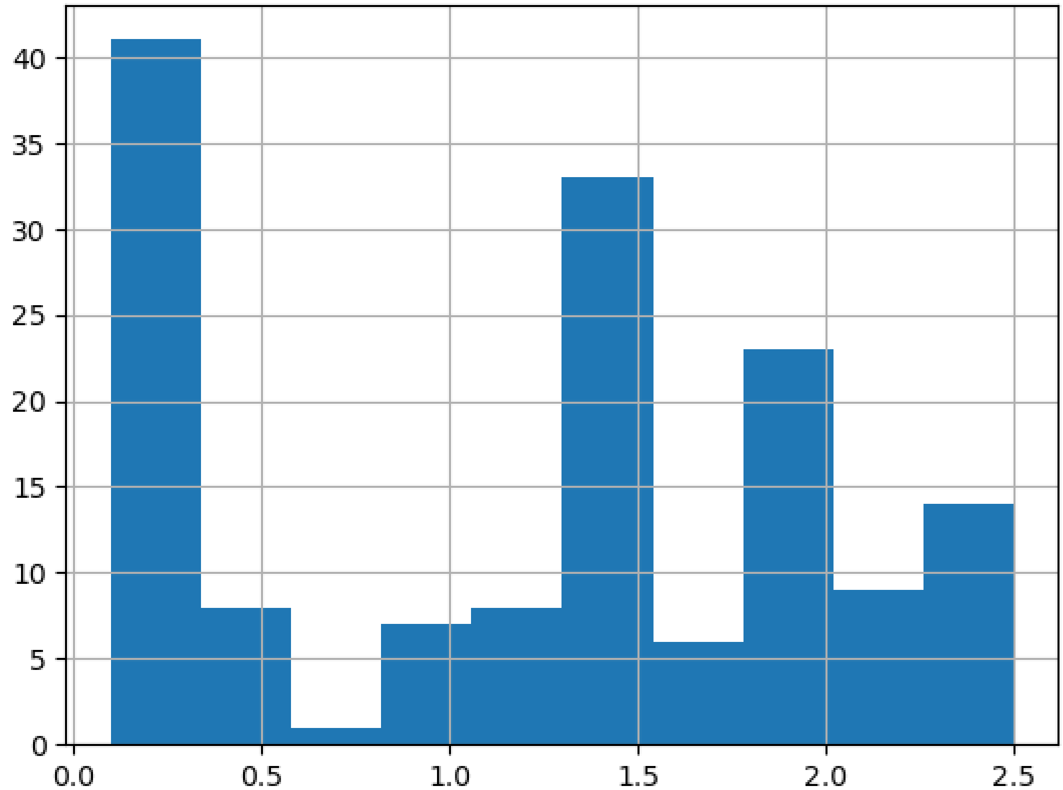
In [14]:
```
df['sepal_width'].hist()
```

Out[14]: <AxesSubplot:>



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

Out[15]: <AxesSubplot:>

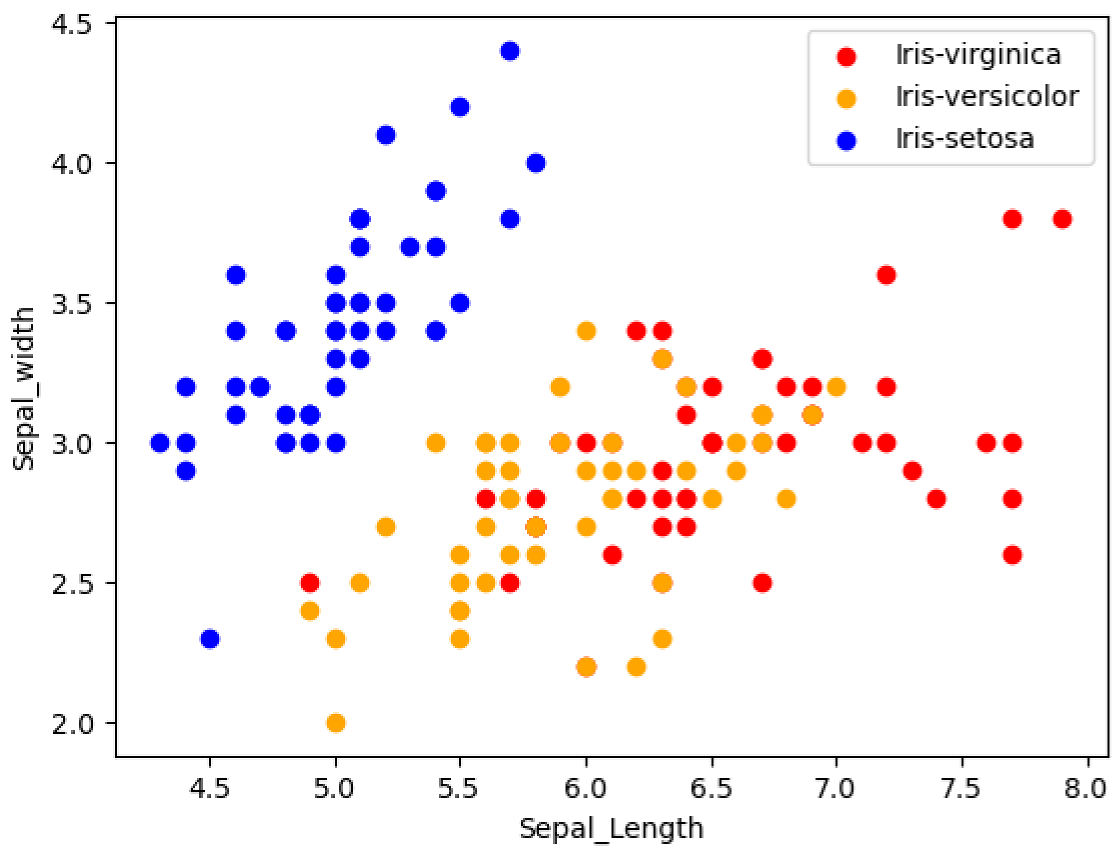


In [17]:
```

#scatter plot
colors = ['red','orange','blue']
species = ['Iris-virginica','Iris-versicolor','Iris-setosa']
```

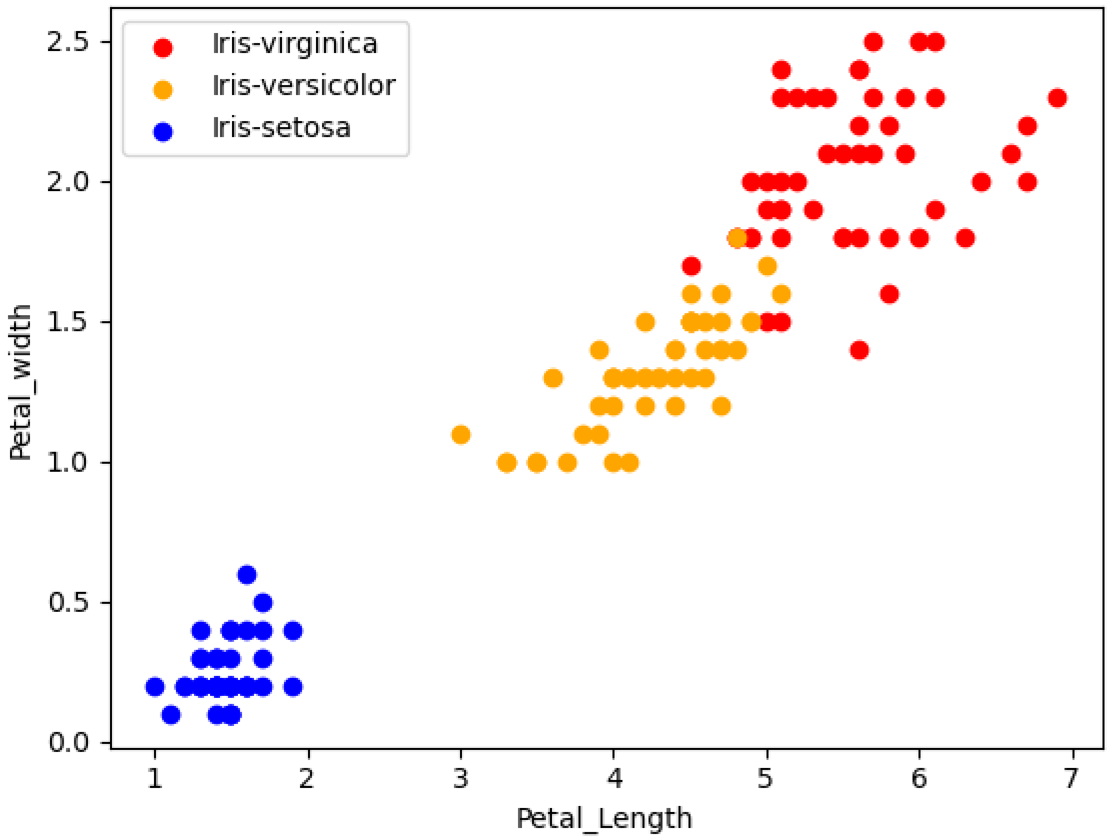
In [18]:
```
for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['sepal_length'],x['sepal_width'], c = colors[i],label = spec
plt.legend()
plt.xlabel('Sepal_Length')
plt.ylabel('Sepal_width')
```

Out[18]:
```
Text(0, 0.5, 'Sepal_width')
```



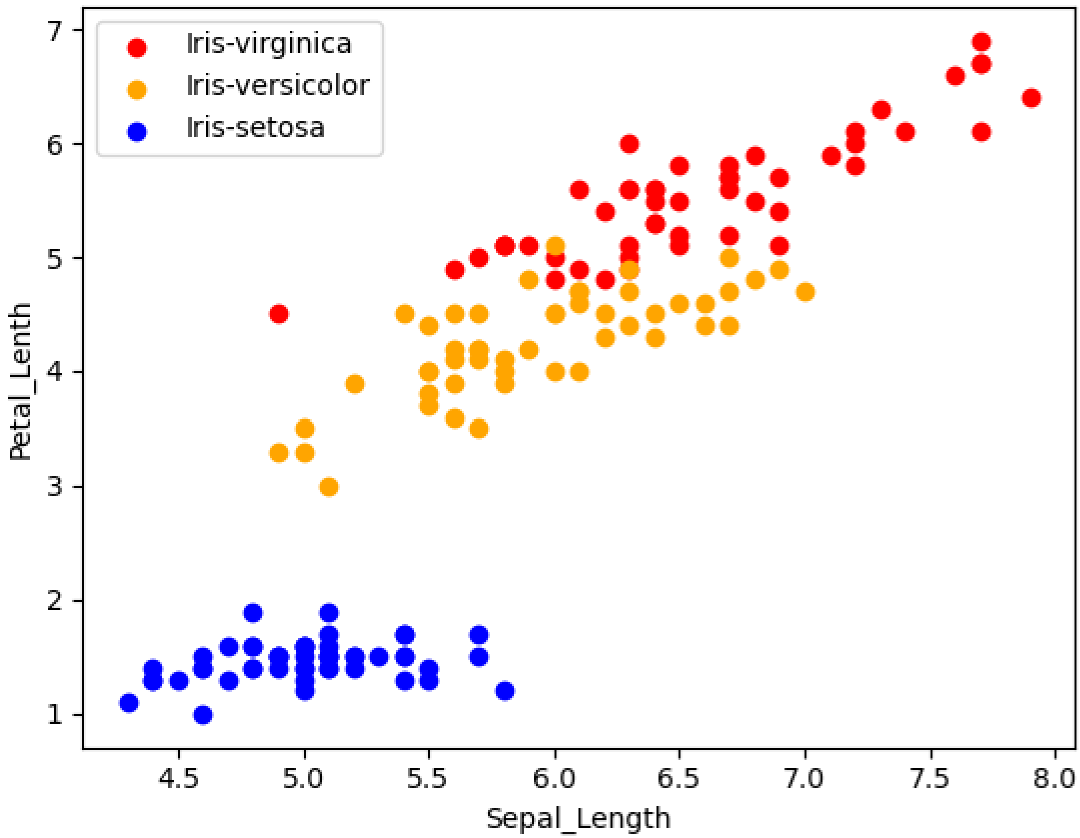
In [19]:
```
for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['petal_length'],x['petal_width'], c = colors[i],label = spec
plt.legend()
plt.xlabel('Petal_Length')
plt.ylabel('Petal_width')
```

Out[19]: Text(0, 0.5, 'Petal_width')



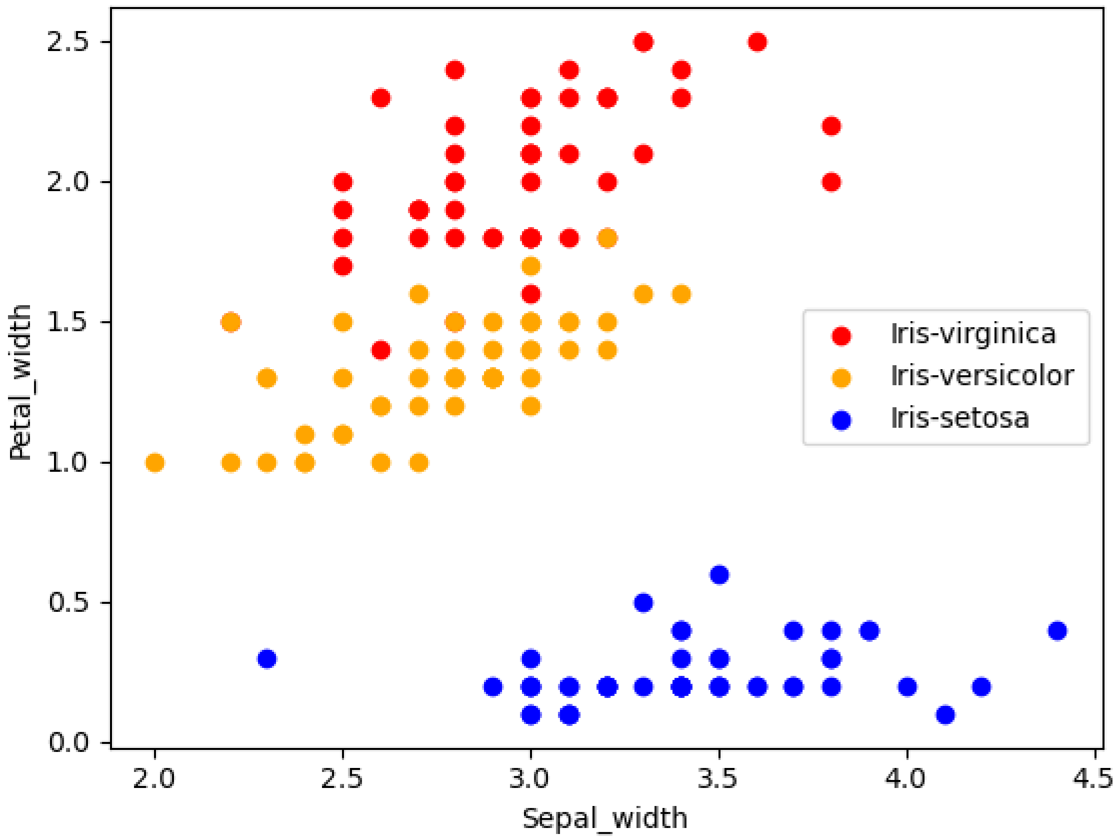
In [20]:
```
for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['sepal_length'],x['petal_length'], c = colors[i],label = spe
plt.legend()
plt.xlabel('Sepal_Length')
plt.ylabel('Petal_Lenth')
```

Out[20]:
```
Text(0, 0.5, 'Petal_Lenth')
```



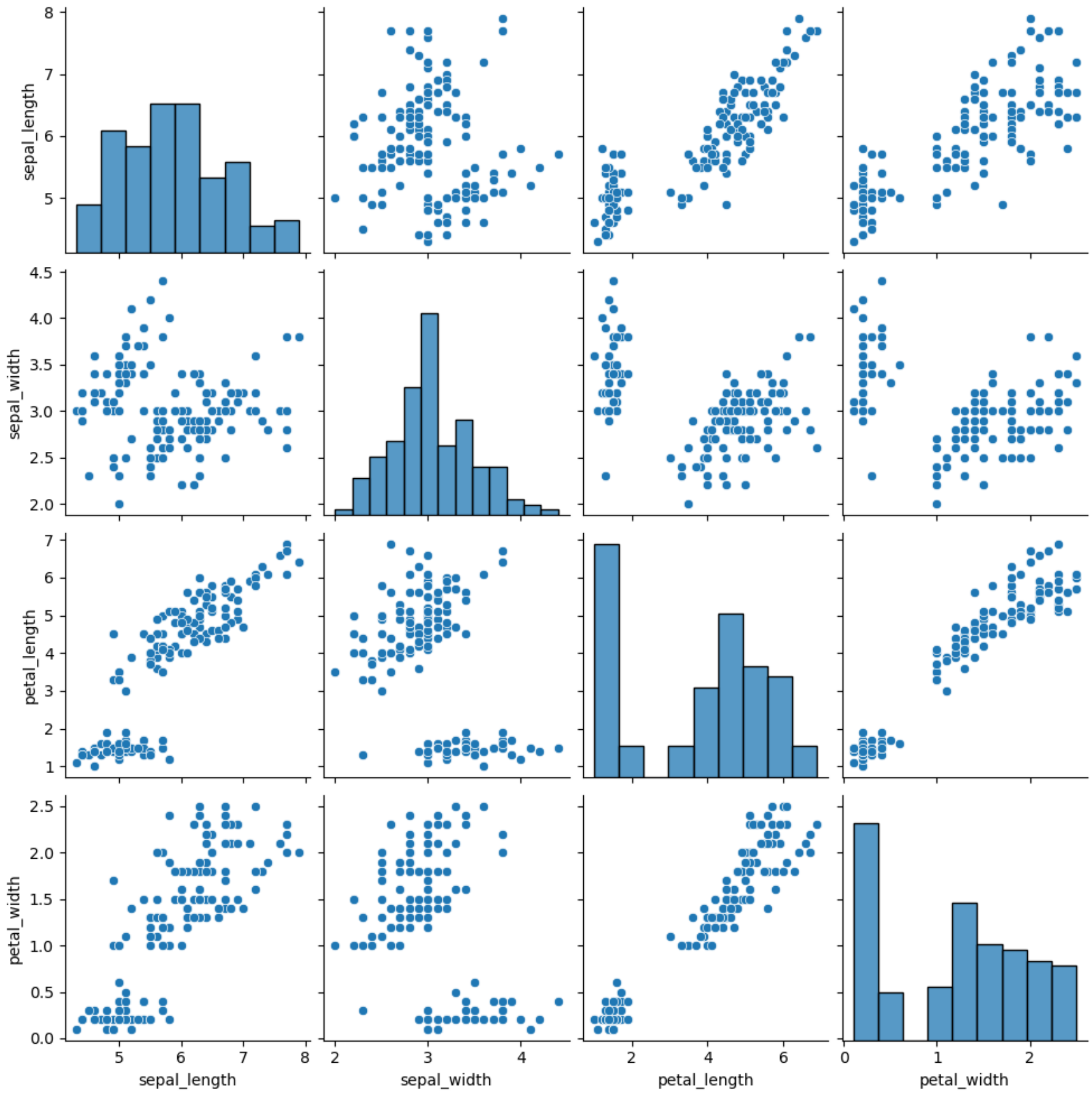
In [21]:
```
for i in range(3):
    x = df[df['species'] == species[i]]
    plt.scatter(x['sepal_width'],x['petal_width'], c = colors[i],label = speci
plt.legend()
plt.xlabel('Sepal_width')
plt.ylabel('Petal_width')
```

Out[21]: Text(0, 0.5, 'Petal_width')



In [22]:
```
sns.pairplot(df)
```

Out[22]: <seaborn.axisgrid.PairGrid at 0x1a235fce7f0>



CORRELATION MATRIX FOR DATASET

A Correlation Matrix is a table showing correlation Coefficient between variables... Each cell in a table shows the Correlation between two Variables...The value is in the range 0 and 1...If two variables have high Correlation, then we can neglect one variable from those two...

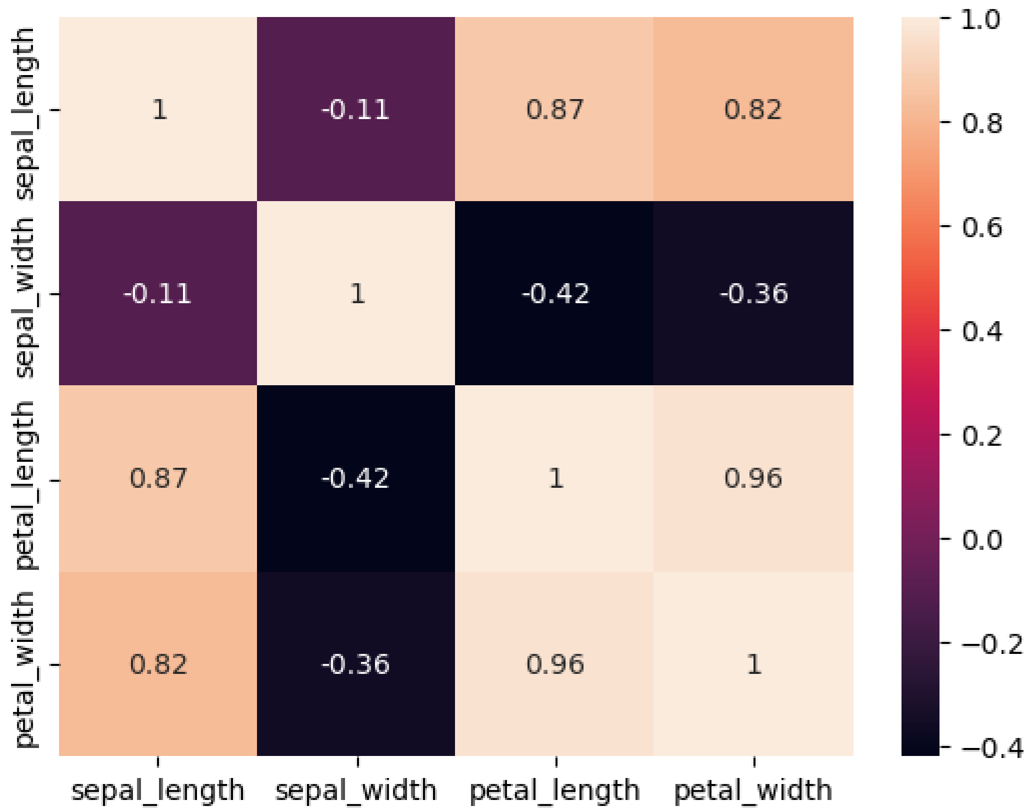
In [23]:
```
df.corr()
```

Out[23]:

	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 [24]:
```
sns.heatmap(df.corr(),annot = True)
```

Out[24]: <AxesSubplot:>



LABEL ENCODER

In ML,We usually deal with dataset which contain multiple labels in one or more than one columns...These labels can be in the form of words or numbers...Label Encoding refers to converting the labels in numerical form... So as to convert it into Machine Readable form

In [25]:
```
import sklearn
from sklearn.preprocessing import LabelEncoder
labelencode = LabelEncoder()
df['species'] = labelencode.fit_transform(df['species'])
```

In [26]:
```
df.head()
```

Out[26]:

	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 [27]:
```
df['species'].unique()
```

Out[27]:
```
array([0, 1, 2])
```

DATA SPLITTING

In [28]:
```
from sklearn.model_selection import train_test_split
```

In [29]:
```
#training the model
#training data = 70%
#testing data = 30%
#splitting data using X and Y varibles
X = df.values[:,0:4]
Y = df.values[:,4]
```

In [30]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.30)
```

USING MODEL ALGORITHMS

K-Nearest-Neighbor Algorithm

In [31]:
```
#IMPORTING KNN
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier()
```

In [32]:
```
model2.fit(X_train,Y_train)
```

Out[32]:
```
KNeighborsClassifier()
```

In [35]:
```
#findinng the accuracy
from warnings import filterwarnings
filterwarnings('ignore')
print("The Accuracy : ",model2.score(X_test,Y_test) * 100)
```

```
The Accuracy :  93.33333333333333
```

In [36]:
```
pre2 = model2.predict(X_test)
from warnings import filterwarnings
filterwarnings('ignore')
```

In [40]:
```
for i in range(len(pre2)):
    print("The given Data is:  ",X_test[i],"The predicted Output is:   ","-->>
#0 indicates Iris-setosa
#1 indicates Iris-Versicolor
#2 indicates Iris-virginica
```

```
The given Data is:   [5.6 3.  4.1 1.3] The predicted Output is:    -->> 1.0
The given Data is:   [5.7 3.8 1.7 0.3] The predicted Output is:    -->> 0.0
The given Data is:   [5.2 3.4 1.4 0.2] The predicted Output is:    -->> 0.0
The given Data is:   [6.3 2.8 5.1 1.5] The predicted Output is:    -->> 2.0
The given Data is:   [5.9 3.2 4.8 1.8] The predicted Output is:    -->> 1.0
The given Data is:   [5.4 3.9 1.7 0.4] The predicted Output is:    -->> 0.0
The given Data is:   [4.9 2.5 4.5 1.7] The predicted Output is:    -->> 1.0
The given Data is:   [5.9 3.  5.1 1.8] The predicted Output is:    -->> 2.0
The given Data is:   [6.4 3.2 5.3 2.3] The predicted Output is:    -->> 2.0
The given Data is:   [6.1 2.8 4.7 1.2] The predicted Output is:    -->> 1.0
The given Data is:   [6.6 3.  4.4 1.4] The predicted Output is:    -->> 1.0
The given Data is:   [5.9 3.  4.2 1.5] The predicted Output is:    -->> 1.0
The given Data is:   [5.7 2.6 3.5 1. ] The predicted Output is:    -->> 1.0
The given Data is:   [4.6 3.4 1.4 0.3] The predicted Output is:    -->> 0.0
The given Data is:   [5.4 3.  4.5 1.5] The predicted Output is:    -->> 1.0
The given Data is:   [4.4 2.9 1.4 0.2] The predicted Output is:    -->> 0.0
The given Data is:   [6.9 3.1 5.4 2.1] The predicted Output is:    -->> 2.0
The given Data is:   [6.8 3.  5.5 2.1] The predicted Output is:    -->> 2.0
The given Data is:   [6.8 2.8 4.8 1.4] The predicted Output is:    -->> 1.0
The given Data is:   [7.  3.2 4.7 1.4] The predicted Output is:    -->> 1.0
The given Data is:   [6.3 2.3 4.4 1.3] The predicted Output is:    -->> 1.0
The given Data is:   [4.7 3.2 1.3 0.2] The predicted Output is:    -->> 0.0
The given Data is:   [5.6 2.7 4.2 1.3] The predicted Output is:    -->> 1.0
The given Data is:   [5.5 2.6 4.4 1.2] The predicted Output is:    -->> 1.0
The given Data is:   [6.9 3.1 5.1 2.3] The predicted Output is:    -->> 2.0
The given Data is:   [5.1 3.3 1.7 0.5] The predicted Output is:    -->> 0.0
The given Data is:   [6.3 2.7 4.9 1.8] The predicted Output is:    -->> 2.0
The given Data is:   [5.8 2.7 5.1 1.9] The predicted Output is:    -->> 2.0
The given Data is:   [5.1 3.5 1.4 0.2] The predicted Output is:    -->> 0.0
The given Data is:   [6.  2.2 5.  1.5] The predicted Output is:    -->> 2.0
The given Data is:   [6.1 2.6 5.6 1.4] The predicted Output is:    -->> 2.0
The given Data is:   [6.7 2.5 5.8 1.8] The predicted Output is:    -->> 2.0
The given Data is:   [6.7 3.1 4.7 1.5] The predicted Output is:    -->> 1.0
The given Data is:   [7.2 3.6 6.1 2.5] The predicted Output is:    -->> 2.0
The given Data is:   [6.  2.9 4.5 1.5] The predicted Output is:    -->> 1.0
The given Data is:   [6.  3.  4.8 1.8] The predicted Output is:    -->> 1.0
The given Data is:   [4.8 3.4 1.6 0.2] The predicted Output is:    -->> 0.0
The given Data is:   [6.3 3.3 6.  2.5] The predicted Output is:    -->> 2.0
The given Data is:   [5.  3.5 1.3 0.3] The predicted Output is:    -->> 0.0
The given Data is:   [5.  3.2 1.2 0.2] The predicted Output is:    -->> 0.0
The given Data is:   [5.1 3.8 1.6 0.2] The predicted Output is:    -->> 0.0
The given Data is:   [6.  2.7 5.1 1.6] The predicted Output is:    -->> 2.0
The given Data is:   [5.1 3.5 1.4 0.3] The predicted Output is:    -->> 0.0
The given Data is:   [5.7 2.5 5.  2. ] The predicted Output is:    -->> 2.0
The given Data is:   [5.4 3.9 1.3 0.4] The predicted Output is:    -->> 0.0
```

CLASSIFICATION REPORT FOR KNN

In [41]:
```
#performing classsification report
from sklearn.metrics import classification_report
print(classification_report(Y_test,pre2))
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        14
         1.0       0.88      0.93      0.90        15
         2.0       0.93      0.88      0.90        16

    accuracy                           0.93        45
   macro avg       0.94      0.94      0.94        45
weighted avg       0.93      0.93      0.93        45
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: