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

Attribute information:

1.speal length in cm

2. sepal width in cm

3. petal length in cm

4. petal width in cm

5. class--> Iris Setosa, Iris Versicolor, Iris Virginica

In [1]:

```
#Import modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

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]:

```
# delete Id
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.to_excel(r'C:\Users\WoU_WSB\Documents\Iris_updated.xlsx', index = False)
```

In [5]:

```
df.to_excel(r'C:\Users\WoU_WSB\Documents\Iris_updated1.xlsx', sheet_name='new', index = Fal
```

In [8]:

```
df.to_csv(r'C:\Users\WoU_WSB\Documents\Iris_updated2.csv', index = False)
```

In [4]:

```
#to display stats about data
df.describe()
```

Out[4]:

	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 [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   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 [6]:

```
df.isnull().sum()
```

Out[6]:

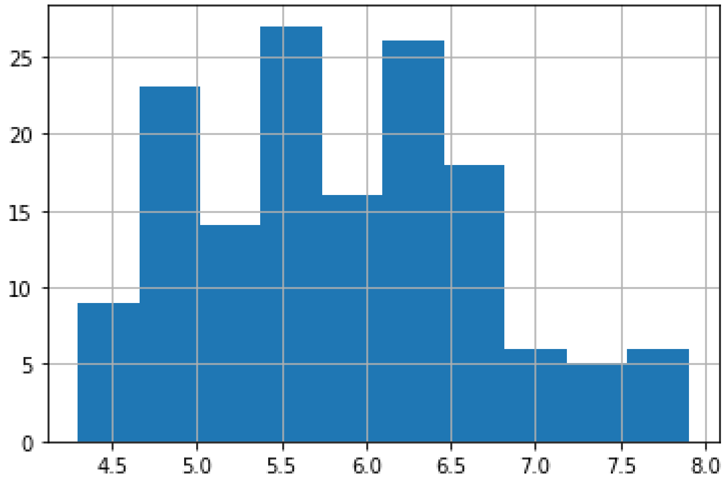
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [7]:

```
#Exploratory data analysis
df['SepalLengthCm'].hist()
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11a68d546d0>
```

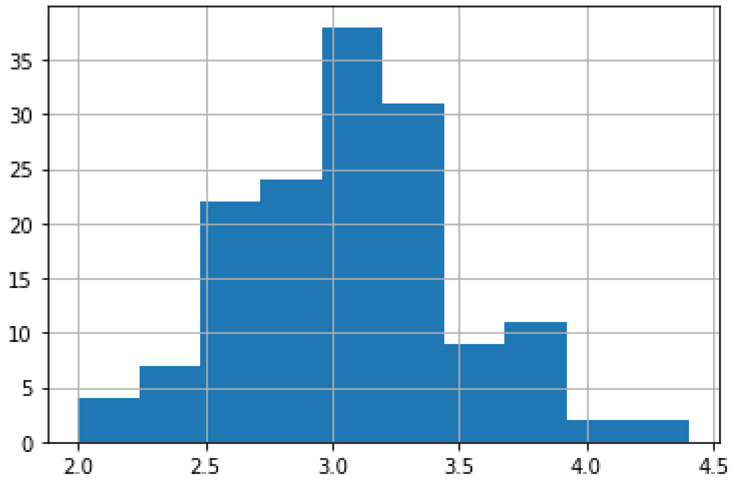


In [8]:

```
df['SepalWidthCm'].hist()
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11a6949fb20>
```

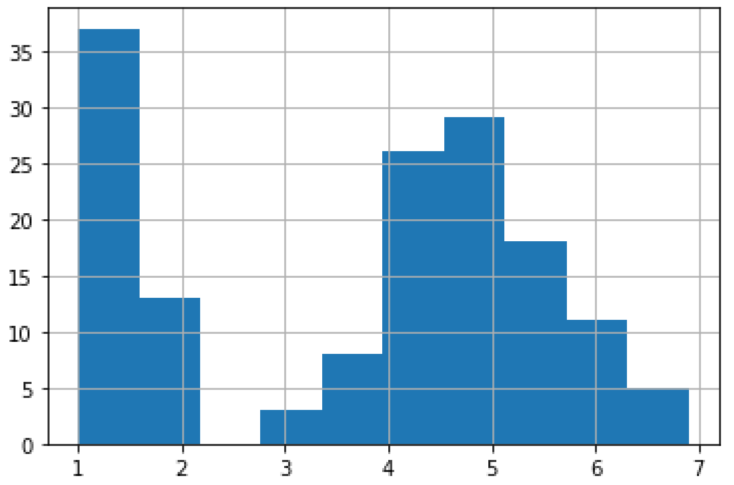


In [9]:

```
df['PetalLengthCm'].hist()
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11a6954b580>
```

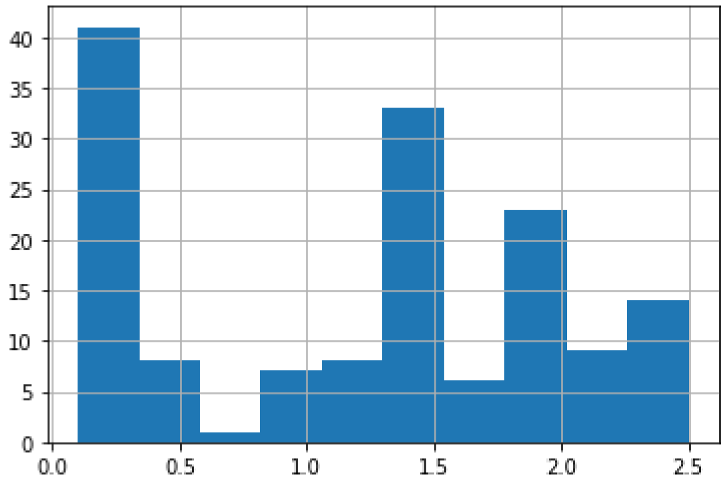


In [10]:

```
df['PetalWidthCm'].hist()
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11a695ced00>
```



In [11]:

```
df['Species'].value_counts()
```

Out[11]:

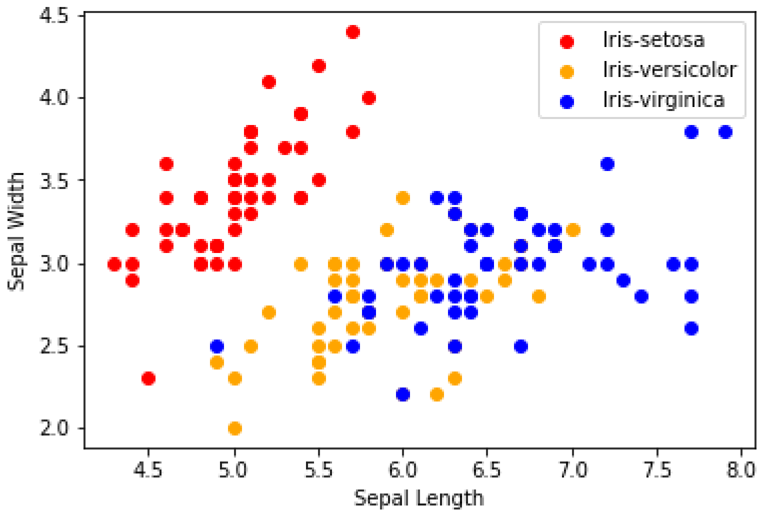
```
Iris-versicolor    50
Iris-setosa        50
Iris-virginica     50
Name: Species, dtype: int64
```

In [12]:

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

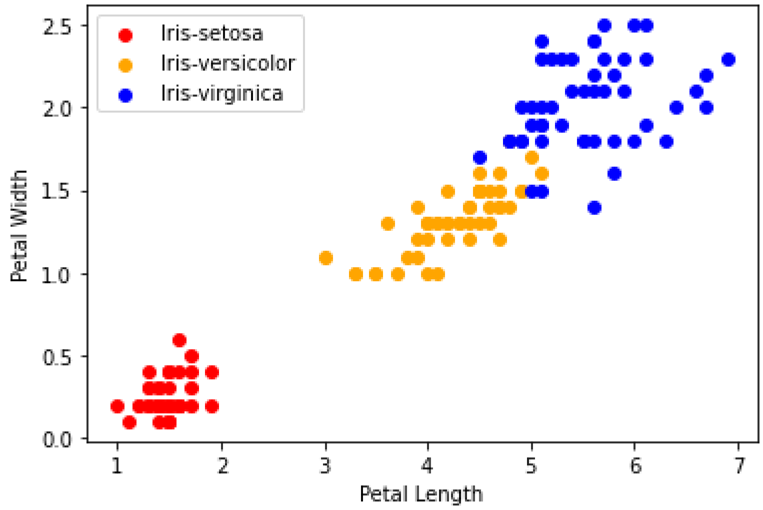
In [13]:

```
#iteration
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()
```



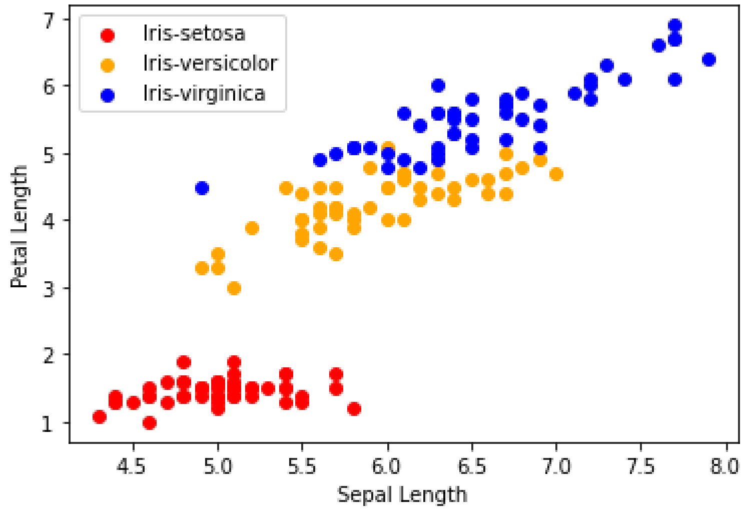
In [14]:

```
#iteration
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()
```



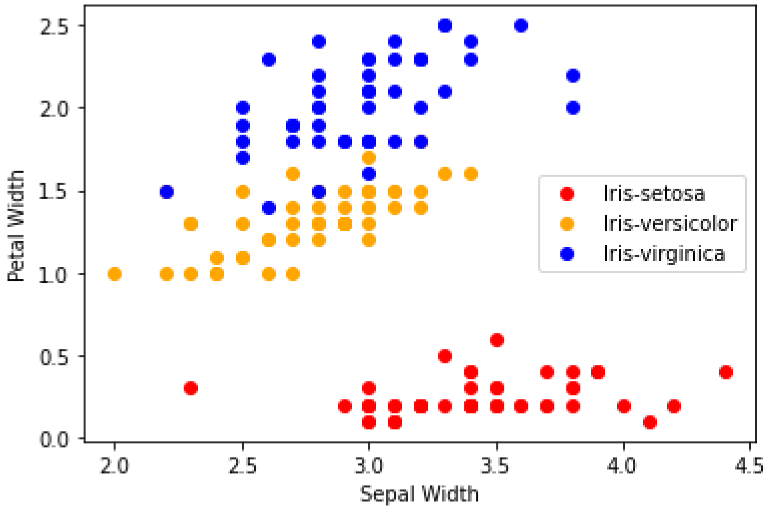
In [15]:

```
#iteration
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("Sepal Length")
    plt.ylabel("Petal Length")
    plt.legend()
```



In [16]:

```
#iteration
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.ylabel("Petal Width")
    plt.legend()
```



In [17]:

```
#Correlation matrix
# value is in the range of -1 to 1

df.corr()
```

Out[17]:

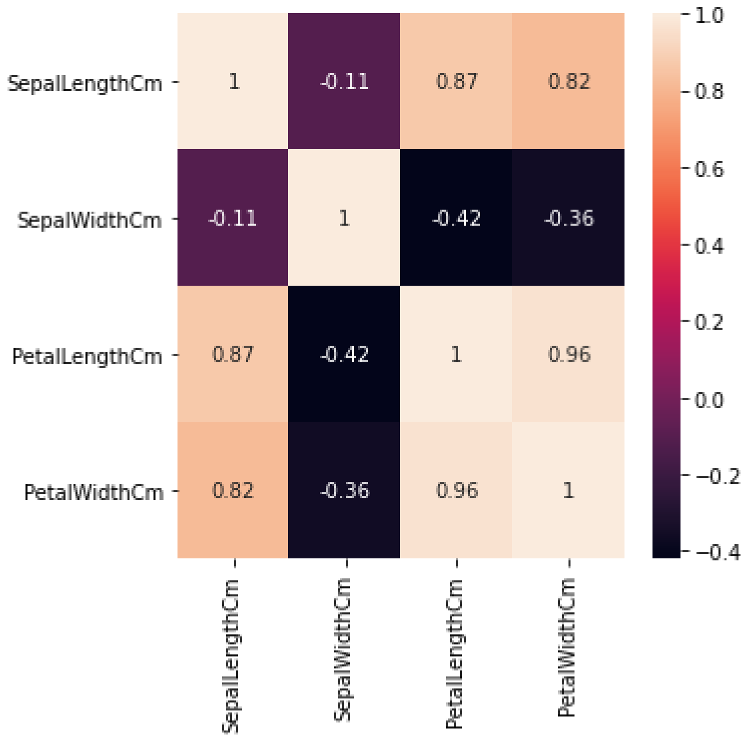
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

In [18]:

```
corr = df.corr()
fig, ax = plt.subplots(figsize=(5,5))
sns.heatmap(corr, annot = True, ax = ax)   # setting annot = true fetches the value form ma
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x11a697bb610>
```



In [19]:

```
# Label Encoder
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
```

In [20]:

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

Out[20]:

	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

In [52]:

```
# split train and test set
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 [53]:

```
# Logistic Regrssion
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train, y_train)
```

Out[53]:

```
LogisticRegression()
```

In [54]:

```
#performance metric
print("Accuracy: ", model.score(x_test, y_test))
```

```
Accuracy:  0.9333333333333333
```

In [55]:

```
#knn algorithm
from sklearn.neighbors import KNeighborsClassifier
model1 = KNeighborsClassifier()
model1.fit(x_train, y_train)
```

Out[55]:

```
KNeighborsClassifier()
```

In [56]:

```
#performance metric
print("Accuracy: ", model1.score(x_test, y_test))
```

```
Accuracy:  0.9555555555555556
```

In [57]:

```
#decision tree
from sklearn.tree import DecisionTreeClassifier
model2 = DecisionTreeClassifier()
model2.fit(x_train, y_train)
```

Out[57]:

```
DecisionTreeClassifier()
```

In [58]:

```
#performance metric
print("Accuracy: ", model2.score(x_test, y_test))
```

```
Accuracy:  0.9333333333333333
```

In [ ]:

In [ ]: