Iris data_set

The data set 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.

Attribute Information:

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- class: -- Iris Setosa -- Iris Versicolour -- Iris Virginica

Import modules

In [1]:
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the dataset

In [141…
```
df=pd.read_csv('Iris.csv')
df.head()
```

Out[141…

	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 [142…
```
#delete the colunm
df=df.drop(columns=['Id'])
```

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

Out[10]:

	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 [12]:
```
#To display the stats about data
df.describe()
```

Out[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

In [15]:
```
#To display basic info of datatype
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 [18]:
```
#To Display no.of sample on each class
df['Species'].value_counts()
```

Out[18]:
```
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: Species, dtype: int64
```

Preprocessing the datset

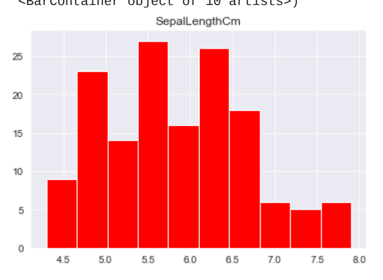
In [22]:
```
#First  of all we check the null values
df.isnull().sum()
```

Out[22]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Exploratory Data Analysis

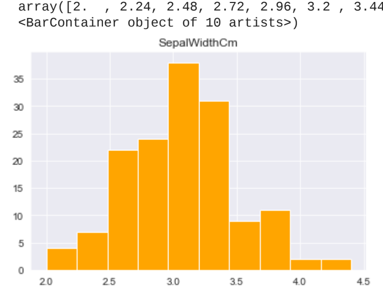
In [49]:
```
plt.title('SepalLengthCm')
plt.hist(df['SepalLengthCm'],color='red')
```

Out[49]:
```
(array([ 9., 23., 14., 27., 16., 26., 18.,  6.,  5.,  6.]),
 array([4.3 , 4.66, 5.02, 5.38, 5.74, 6.1 , 6.46, 6.82, 7.18, 7.54, 7.9 ]),
 <BarContainer object of 10 artists>)
```



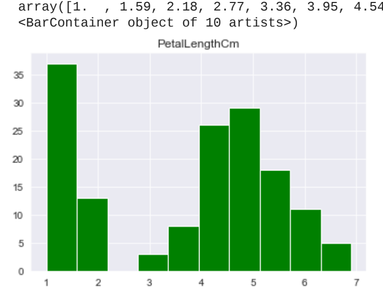
In [50]:
```
plt.title('SepalWidthCm')
plt.hist(df['SepalWidthCm'],color='orange')
```

Out[50]:
```
(array([ 4.,  7., 22., 24., 38., 31.,  9., 11.,  2.,  2.]),
 array([2.  , 2.24, 2.48, 2.72, 2.96, 3.2 , 3.44, 3.68, 3.92, 4.16, 4.4 ]),
 <BarContainer object of 10 artists>)
```



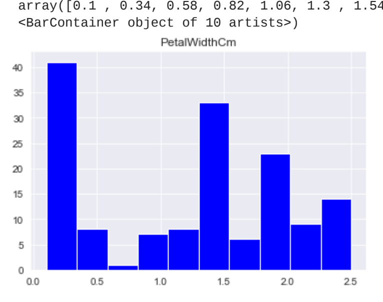
In [48]:
```
plt.title('PetalLengthCm')
plt.hist(df['PetalLengthCm'],color='Green')
```

Out[48]:
```
(array([37., 13.,  0.,  3.,  8., 26., 29., 18., 11.,  5.]),
 array([1.  , 1.59, 2.18, 2.77, 3.36, 3.95, 4.54, 5.13, 5.72, 6.31, 6.9 ]),
 <BarContainer object of 10 artists>)
```



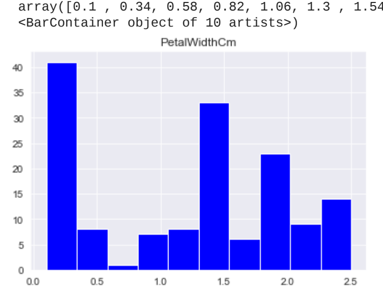
In [45]:
```
plt.title('PetalWidthCm')
plt.hist(df['PetalWidthCm'],color='blue')
```

Out[45]:
```
(array([41.,  8.,  1.,  7.,  8., 33.,  6., 23.,  9., 14.]),
 array([0.1 , 0.34, 0.58, 0.82, 1.06, 1.3 , 1.54, 1.78, 2.02, 2.26, 2.5 ]),
 <BarContainer object of 10 artists>)
```



In [84]:
```
plt.title('PetalWidthCm')
plt.hist(df['PetalWidthCm'],color='blue')
```

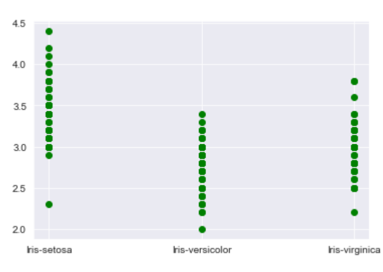
Out[84]:
```
(array([41.,  8.,  1.,  7.,  8., 33.,  6., 23.,  9., 14.]),
 array([0.1 , 0.34, 0.58, 0.82, 1.06, 1.3 , 1.54, 1.78, 2.02, 2.26, 2.5 ]),
 <BarContainer object of 10 artists>)
```



In [ ]:

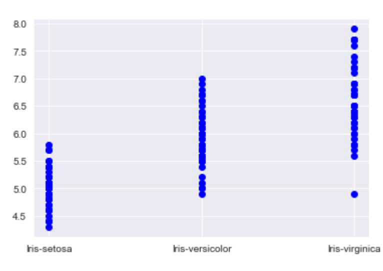
In [71]:
```
plt.scatter(df['Species'],df['SepalWidthCm'],color='green')
```

Out[71]:
```
<matplotlib.collections.PathCollection at 0x1405998f820>
```



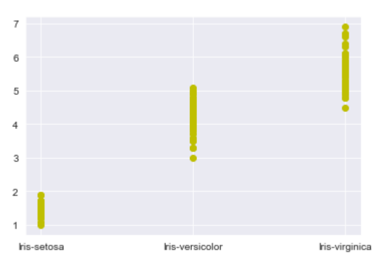
In [86]:
```
plt.scatter(df['Species'],df['SepalLengthCm'],color='b')
```

Out[86]:
```
<matplotlib.collections.PathCollection at 0x14058793400>
```



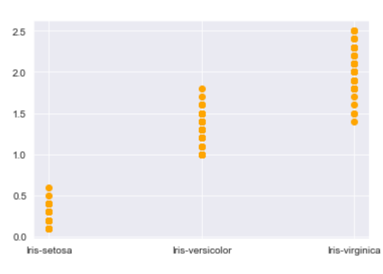
In [87]:
```
plt.scatter(df['Species'],df['PetalLengthCm'],color='y')
```

Out[87]:
```
<matplotlib.collections.PathCollection at 0x140580f7eb0>
```



In [89]:
```
plt.scatter(df['Species'],df['PetalWidthCm'],color='orange')
```

Out[89]:
```
<matplotlib.collections.PathCollection at 0x140554cc4c0>
```



Coorelation Matrix

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

In [97]:
```
import numpy as np
np.corrcoef(df['SepalLengthCm'],df['SepalWidthCm'])
```

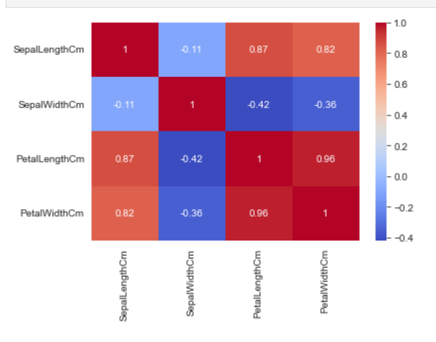
Out[97]:
```
array([[ 1.        , -0.10936925],
       [-0.10936925,  1.        ]])
```

In [98]:
```
df.corr()
```

Out[98]:

	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 [106…
```
dataplot = sns.heatmap(df.corr(), cmap="coolwarm", annot=True)
```



Label Encoder

In machine learning, we usually deal with datasets which contains 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 into numeric form so as to convert it into the machine-readable form

In [143…
```
df.head()
```

Out[143…

	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 [144…
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [145…
```
df['Species']=le.fit_transform(df['Species'])
df.head()
```

Out[145…

	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 [146…
```
df['Species'].unique
```

Out[146…
```
<bound method Series.unique of 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>
```

Select the Label and Features

In [147…
```
Y=df['Species']
X=df.drop(columns=['Species'])
# x #Features
```

In [148…
```
Y #Label
```

Out[148…
```
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
```

Splitting a dataset

In [149…
```
from sklearn.model_selection import train_test_split
# train=80
#test=20
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.20,random_state=20)
```

In [150…
```
len(x_train),len(y_train)
```

Out[150…
```
(120, 120)
```

In [151…
```
#Logistic Regression
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

In [152…
```
model.fit(x_train,y_train)
```

Out[152…
```
LogisticRegression()
```

In [156…
```
#Model accuracy
print('Accuracy: ',model.score(x_test,y_test))
```

```
Accuracy:  0.9333333333333333
```

In [ ]: