LETSGROWMORE

BEGINNER LEVEL task 1

Iris Flowers Classification ML Project

In [1]:
```
from PIL import Image
```

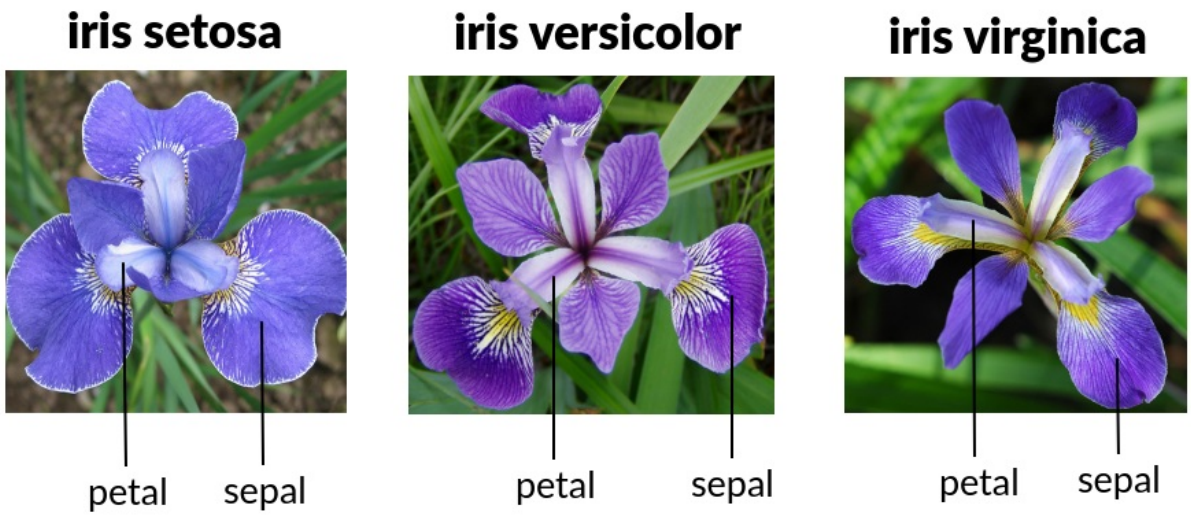
In [3]:
```
irf=Image.open('irisimages.png')
```

In [4]:
```
print(irf)
```

```
<PIL.PngImagePlugin.PngImageFile image mode=RGB size=1000x447 at 0x2235EB16550>
```

In [5]:
```
irf
```

Out[5]:



In [6]:
```
#importing the libraries
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
```

In [10]:
```
#load and read the iris dataset
data=pd.read_csv('irisdata.csv')
```

In [11]:
```
print(data)
```

```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
```

```
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

In [12]:
```
data.head()
```

Out[12]:

	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 [13]:
```
data.tail()
```

Out[13]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [14]:
```
data.head(10)
```

Out[14]:

	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 [15]:
```
data=data.drop(columns=['Id'])
```

In [16]:
```
data.head()
```

Out[16]:

	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 [17]:
```
data.shape
```

Out[17]:
```
(150, 5)
```

In [20]:
```
data.value_counts()
```

Out[20]:
```
SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
4.9            3.1           1.5            0.1           Iris-setosa       3
5.8            2.7           5.1            1.9           Iris-virginica    2
               4.0           1.2            0.2           Iris-setosa       1
5.9            3.0           4.2            1.5           Iris-versicolor   1
6.2            3.4           5.4            2.3           Iris-virginica    1
                                                                            ..
5.5            2.3           4.0            1.3           Iris-versicolor   1
               2.4           3.7            1.0           Iris-versicolor   1
                             3.8            1.1           Iris-versicolor   1
               2.5           4.0            1.3           Iris-versicolor   1
7.9            3.8           6.4            2.0           Iris-virginica    1
Length: 147, dtype: int64
```

exploratory data analysis

In [22]:
```
data.isnull()
```

Out[22]:

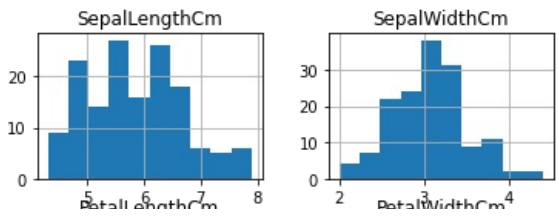
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

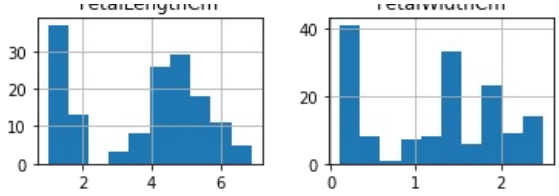
150 rows × 5 columns

In [23]:
```
data.isnull().sum()
```

Out[23]:
```
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

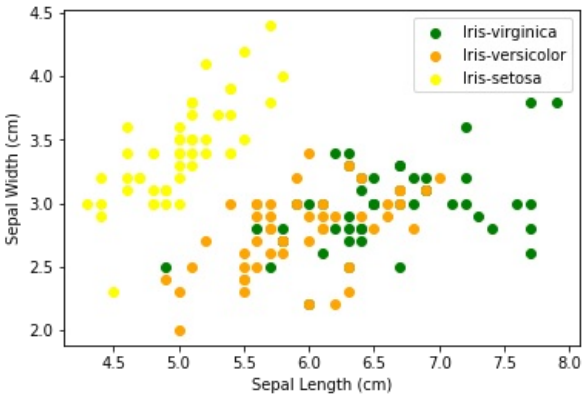
In [32]:
```
data.hist()
plt.show()
```





In [36]:
```
c=['green', 'orange', 'yellow']
f=['Iris-virginica', 'Iris-versicolor', 'Iris-setosa']
for i in range(3):
    x=data[data['Species'] == f[i]]
    plt.scatter(x['SepalLengthCm'], x['SepalWidthCm'], c=c[i], label=f[i])
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.legend()
```

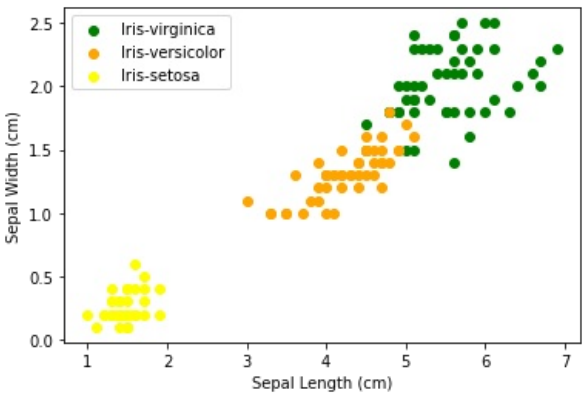
Out[36]: <matplotlib.legend.Legend at 0x22324388f40>



the iris-setosa is completely remains separate with iris virginica and iris versicolor and in the above visualisation can see the correlation between Versicolor and virginica

In [37]:
```
#similarly
c=['green', 'orange', 'yellow']
f=['Iris-virginica', 'Iris-versicolor', 'Iris-setosa']
for i in range(3):
    x=data[data['Species'] == f[i]]
    plt.scatter(x['PetalLengthCm'], x['PetalWidthCm'], c=c[i], label=f[i])
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.legend()
```

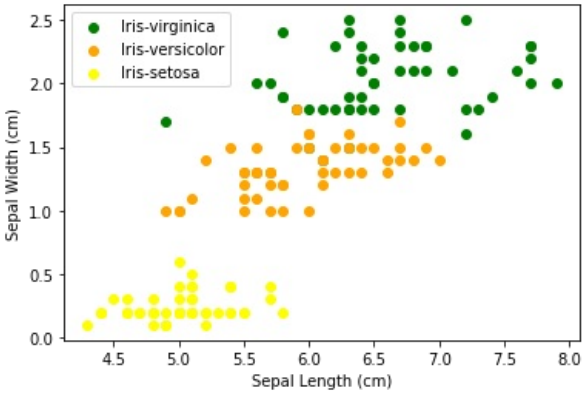
Out[37]: <matplotlib.legend.Legend at 0x22324419910>



In [38]:
```
c=['green', 'orange', 'yellow']
f=['Iris-virginica', 'Iris-versicolor', 'Iris-setosa']
for i in range(3):
    x=data[data['Species'] == f[i]]
    plt.scatter(x['SepalLengthCm'], x['PetalWidthCm'], c=c[i], label=f[i])
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
```

```
plt.legend()
```

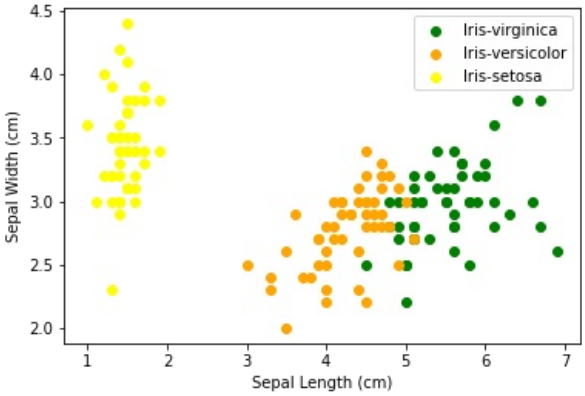
Out[38]: <matplotlib.legend.Legend at 0x223244690d0>



In [39]:

```
c=['green', 'orange', 'yellow']
f=['Iris-virginica', 'Iris-versicolor', 'Iris-setosa']
for i in range(3):
    x=data[data['Species'] == f[i]]
    plt.scatter(x['PetalLengthCm'], x['SepalWidthCm'], c=c[i], label=f[i])
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.legend()
```

Out[39]: <matplotlib.legend.Legend at 0x22324288850>



hence we can observed from our dataset the iris setosa is completely isolated from iris-virginica and iris versinicolor

In [41]:

```
data.corr()
```

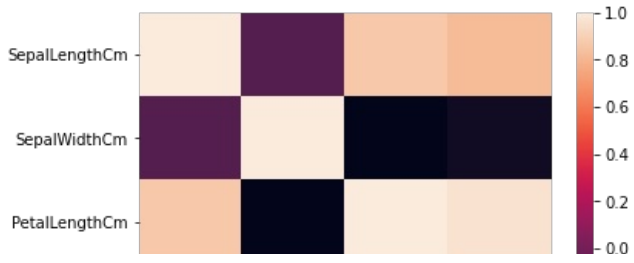
Out[41]:

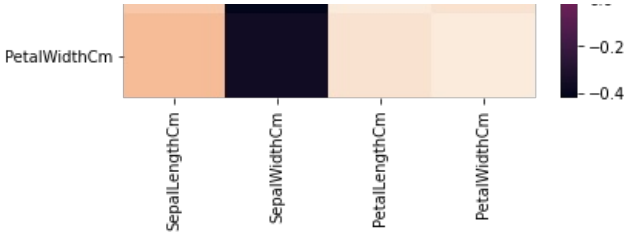
	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
SepalLengthCm	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.817954	-0.356544	0.962757	1.000000

In [42]:

```
sns.heatmap(data.corr())
```

Out[42]: <AxesSubplot:>





importing the machine learning libraries

In [50]:
```
from sklearn.linear_model import LinearRegression,LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
```

convert the attributes values in machine readable form we have species values with string

In [51]:
```
e=LabelEncoder()
data['Species']=e.fit_transform(data['Species'])
```

In [52]:
```
data.head(10)
```

Out[52]:

	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
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5.0	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0

In [53]:
```
data.tail()
```

Out[53]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

the values are replced with 0,1 and 2 because we have three species in dataset

model training

In [54]:
```
#we need to seperate the target value from iris dataset
```

In [55]:
```
x=data.drop(columns=['Species'])
y=data['Species']
```

In [56]:
```
x
```

Out[56]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

In [57]:
```
y
```

Out[57]:
```
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 [78]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
```

using LogisticsRegression

In [79]:
```
lg=LogisticRegression()
```

In [84]:
```
#fitting the model with training dataset
lg.fit(x_train,y_train)
```

```
C:\Users\KORRA SRINU\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarning: lbfgs
failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[84]:
```
LogisticRegression()
```

In [94]:
```
#after fitting the train data printing the accuracy of the data
#model score
s=lg.score(x_test,y_test)*100
```

In [95]:
```
print(s)
```

```
95.55555555555556
```

In [96]:
```
print("the accuracy of logistic regression is ",s)
```

```
the accuracy of logistic regression is  95.55555555555556
```

In [106…
```
y_predict=lg.predict(x_test)
```

In [107…
```
print(y_predict)
```

```
[2 2 2 0 2 1 0 1 0 0 1 1 1 0 2 2 0 1 2 0 0 0 0 0 0 2 1 2 1 0 0 0 1 1 0 2 2
 2 0 1 0 1 2 2 0]
```

using decision tree model

In [97]:
```
#using decision Tree Model
dt=DecisionTreeClassifier()
```

In [98]:
```
#fit the train data
dt.fit(x_train,y_train)
```

Out[98]:
```
DecisionTreeClassifier()
```

In [99]:
```
dscore=dt.score(x_test,y_test)*100
```

In [100…
```
print(dscore)
```

```
93.33333333333333
```

In [101…
```
print("the accuracy of Decision tree model is",dscore)
```

```
the accuracy of Decision tree model is 93.33333333333333
```

In [109…
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_predict)
print(cm)
```

```
[[19  0  0]
 [ 0 11  1]
 [ 0  1 13]]
```

The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier.

In [ ]:
```

```