Iris Flower classification

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [ ]:

```
df=pd.read_csv('Iris.csv',index_col=0)
df.head()
```

Out[ ]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
Id					
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 150 entries, 1 to 150
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: 7.0+ KB
```

In [ ]:

```
#checking for null values
df.isnull().sum()
```

Out[ ]:

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

In [ ]:

```
#output labels
df['Species'].unique()
```

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

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

Out[ ]:

	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

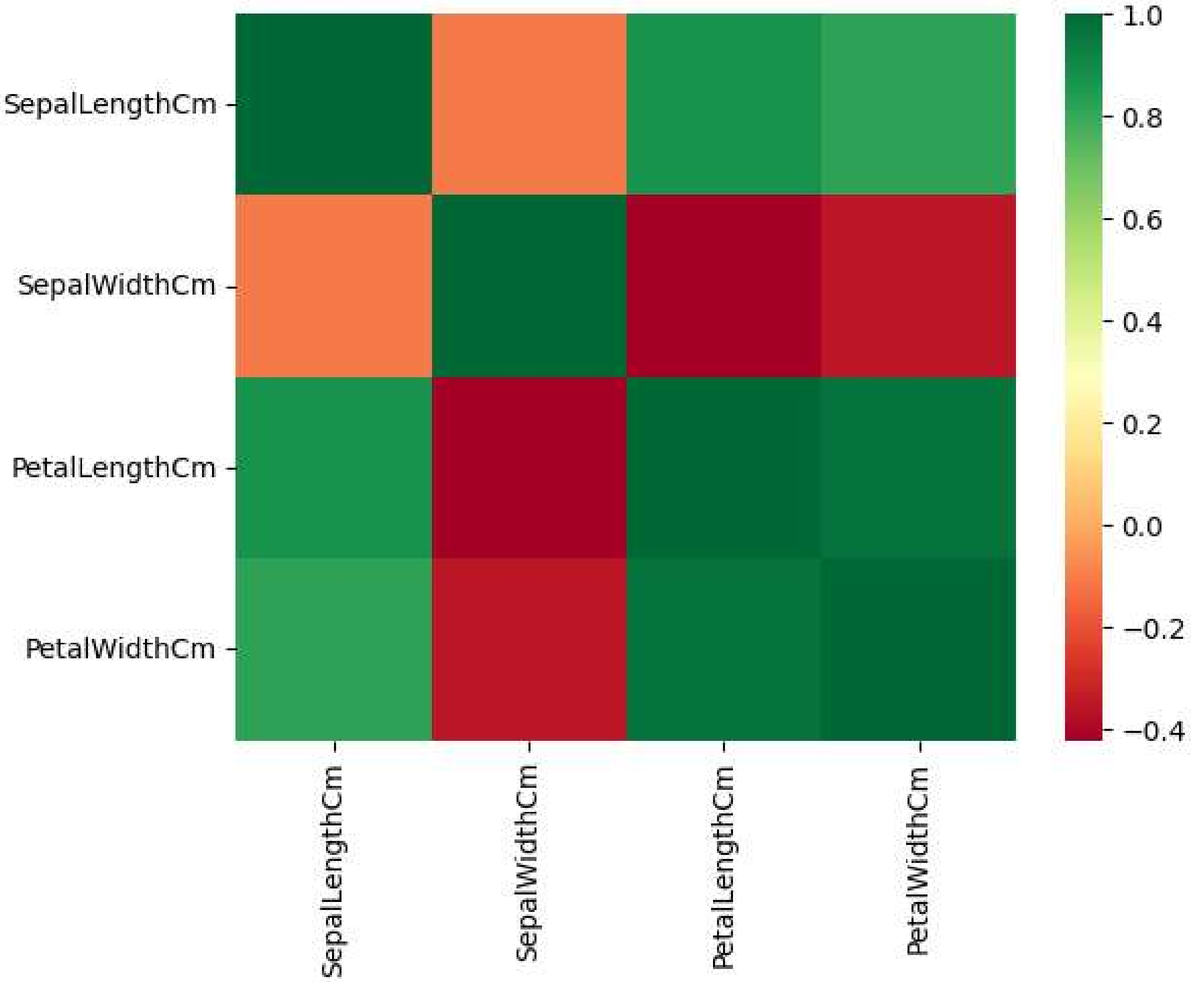
In [ ]:
```
#correlation
df.iloc[:,:4].corr()
```

Out[ ]:

	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

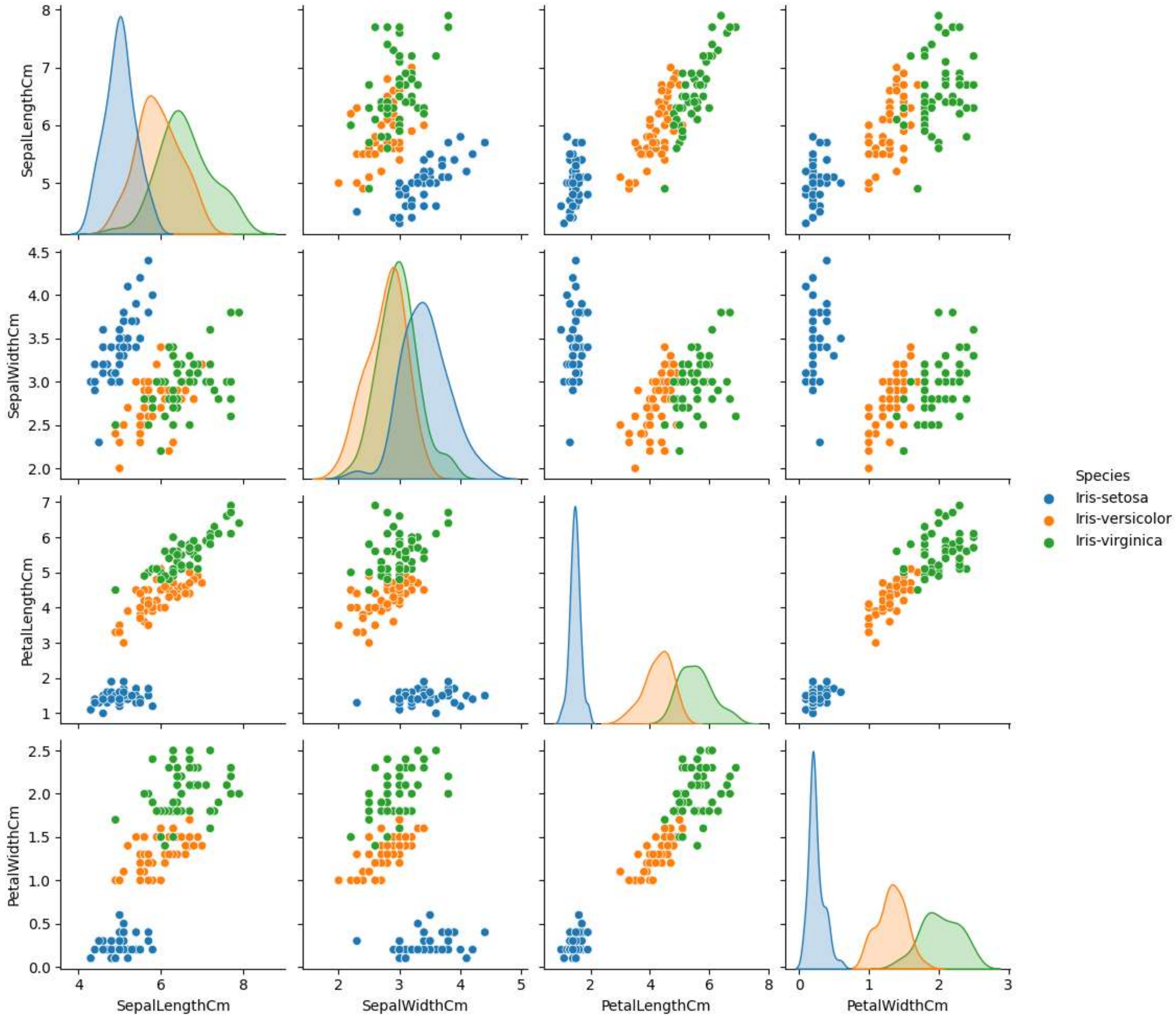
In [ ]:
```
#heatmap
sns.heatmap(df.iloc[:,:4].corr(),cmap="RdYlGn")
```

Out[ ]:
```
<Axes: >
```



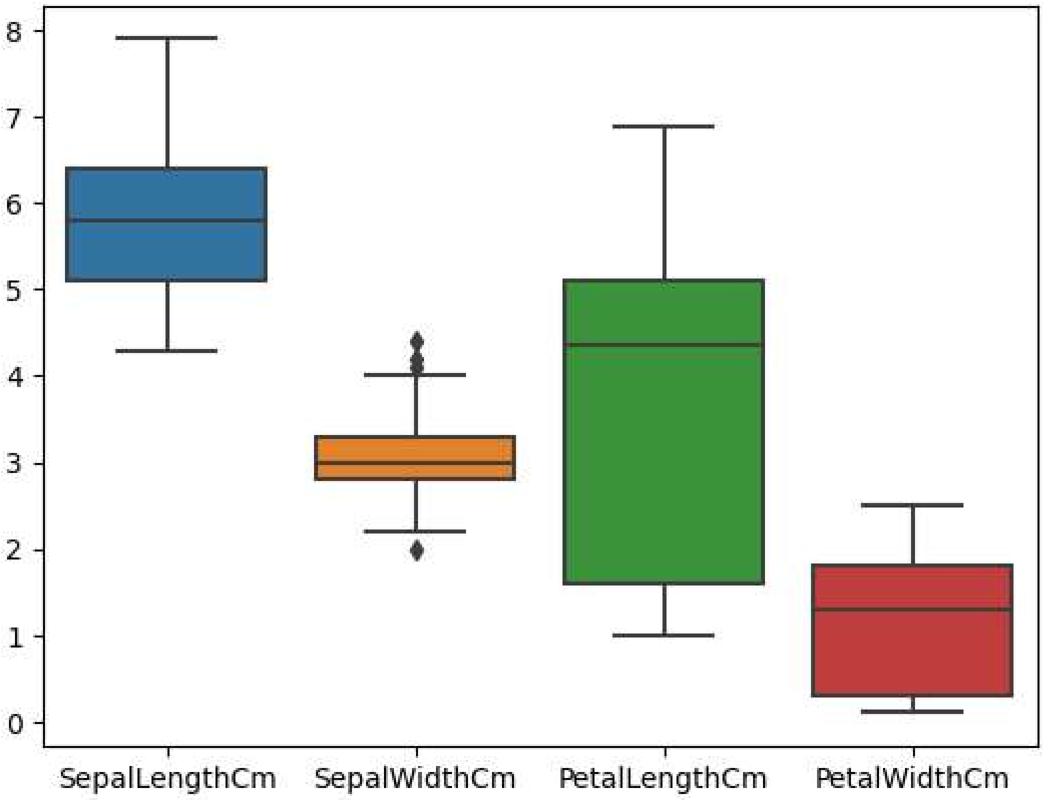
```
In [ ]: sns.pairplot(df,hue='Species')
```

```
Out[ ]: <seaborn.axisgrid.PairGrid at 0x25c4d20b950>
```



```
In [ ]: sns.boxplot(df)
```

```
Out[ ]: <Axes: >
```



In [ ]:

```
#final dataset
data=df.values
x=data[:,:4]
y=data[:,4]
```

In [ ]:

```
print(x[:10])
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.0 1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.0 3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.0 3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]]
```

In [ ]:

```
print(y[:10])
```

```
['Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa']
```

In [ ]:

```
#splitting data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [ ]:

```
print(y_test)
```

```
['Iris-setosa' 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-virginica'
 'Iris-setosa' 'Iris-versicolor' 'Iris-virginica' 'Iris-versicolor'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-setosa' 'Iris-versicolor'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica']
```

In [ ]:

```
#logistic regression model
from sklearn.linear_model import LogisticRegression
logr_model=LogisticRegression()
logr_model.fit(x_train,y_train)
```

Out[ ]:

```
▾ LogisticRegression

LogisticRegression()
```

In [ ]:

```
#testing logistic regression model
logr_predict=logr_model.predict(x_test)

for i in range(len(logr_predict)):
    print(y_test[i],logr_predict[i])
```

```
Iris-setosa Iris-setosa
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-setosa Iris-setosa
Iris-versicolor Iris-versicolor
Iris-versicolor Iris-versicolor
Iris-virginica Iris-virginica
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-versicolor Iris-versicolor
Iris-setosa Iris-setosa
Iris-virginica Iris-virginica
Iris-setosa Iris-setosa
Iris-versicolor Iris-versicolor
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-setosa Iris-setosa
Iris-setosa Iris-setosa
Iris-setosa Iris-setosa
Iris-setosa Iris-setosa
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-versicolor Iris-versicolor
Iris-setosa Iris-setosa
Iris-versicolor Iris-versicolor
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-virginica Iris-virginica
Iris-virginica Iris-virginica
Iris-virginica Iris-virginica
```

In [ ]:

```
#accuracy
from sklearn.metrics import accuracy_score,classification_report
logr_acc=accuracy_score(y_test,logr_predict)
logr_report=classification_report(y_test,logr_predict)
print(f'logistic regression \naccuracy={logr_acc*100}\n{logr_report}')
```

```
logistic regression
accuracy=100.0
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         9
Iris-versicolor       1.00      1.00      1.00        11
 Iris-virginica       1.00      1.00      1.00        10

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

In [ ]:

```
#svm classifier
from sklearn.svm import SVC
svc_model=SVC()
svc_model.fit(x_train,y_train)
```

Out[ ]:

```
▾ SVC
SVC()
```

In [ ]:

```
svm_predict=svc_model.predict(x_test)
for i in range(len(svm_predict)):
    print(y_test[i],svm_predict[i])
```

```
Iris-setosa Iris-setosa
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-setosa Iris-setosa
Iris-versicolor Iris-versicolor
Iris-versicolor Iris-versicolor
Iris-virginica Iris-virginica
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-versicolor Iris-versicolor
Iris-setosa Iris-setosa
Iris-virginica Iris-virginica
Iris-setosa Iris-setosa
Iris-versicolor Iris-versicolor
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-setosa Iris-setosa
Iris-setosa Iris-setosa
Iris-setosa Iris-setosa
Iris-setosa Iris-setosa
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-versicolor Iris-versicolor
Iris-setosa Iris-setosa
Iris-versicolor Iris-versicolor
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-virginica Iris-virginica
Iris-virginica Iris-virginica
Iris-virginica Iris-virginica
```

In [ ]:

```
#accuracy
from sklearn.metrics import accuracy_score,classification_report
svm_acc=accuracy_score(y_test,svm_predict)
svm_report=classification_report(y_test,svm_predict)
print(f'svm classifier \naccuracy={svm_acc*100}\n{svm_report}')
```

```
svm classifier
accuracy=100.0
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         9
Iris-versicolor       1.00      1.00      1.00        11
 Iris-virginica       1.00      1.00      1.00        10

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

In [ ]:

```
#decisiontree classifier
from sklearn.tree import DecisionTreeClassifier
model3=DecisionTreeClassifier()
model3.fit(x_train,y_train)
```

Out[ ]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [ ]:

```
d_predict=model3.predict(x_test)
for i in range(len(d_predict)):
    print(y_test[i],d_predict[i])
```

```
Iris-setosa Iris-setosa
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-setosa Iris-setosa
Iris-versicolor Iris-versicolor
Iris-versicolor Iris-versicolor
Iris-virginica Iris-virginica
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-versicolor Iris-versicolor
Iris-setosa Iris-setosa
Iris-virginica Iris-virginica
Iris-setosa Iris-setosa
Iris-versicolor Iris-versicolor
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-setosa Iris-setosa
Iris-setosa Iris-setosa
Iris-setosa Iris-setosa
Iris-setosa Iris-setosa
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-versicolor Iris-versicolor
Iris-setosa Iris-setosa
Iris-versicolor Iris-versicolor
Iris-virginica Iris-virginica
Iris-versicolor Iris-versicolor
Iris-virginica Iris-versicolor
Iris-virginica Iris-virginica
Iris-virginica Iris-virginica
```

In [ ]:

```
#accuracy
from sklearn.metrics import accuracy_score,classification_report
d_acc=accuracy_score(y_test,d_predict)
d_report=classification_report(y_test,d_predict)
print(f'DCT classifier \naccuracy={d_acc*100}\n{d_report}')
```

```
DCT classifier
accuracy=96.66666666666667
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         9
Iris-versicolor       0.92      1.00      0.96        11
 Iris-virginica       1.00      0.90      0.95        10

       accuracy                           0.97        30
      macro avg       0.97      0.97      0.97        30
   weighted avg       0.97      0.97      0.97        30
```

By the end #Logistic regression classifier and #svm classifier performed well

In [ ]:

```
#Test with own cases by #Logistic regression classifier
input=[[5.1,3.8,1.5,0.3],[5.5,2.3,4.0,1.3],[6.0,2.2,5.0,1.5],[5.8,2.8,5.1,2.4]]
output=[ 'Iris-setosa','Iris-versicolor','Iris-virginica','Iris-virginica']
predict=logr_model.predict(input)
for i in range(len(predict)):
    print(output[i],predict[i])
```

```
Iris-setosa Iris-setosa
Iris-versicolor Iris-versicolor
Iris-virginica Iris-virginica
Iris-virginica Iris-virginica
```

In [ ]:

```
import pickle
fi=open('irsis_Logistic regression classifier.pkl','wb')
pickle.dump(logr_model,fi)
```