IRIS FLOWER CLASSIFICATION

Iris flower classification is a very popular machine learning project. The iris dataset contains three classes of flowers, Versicolor, Setosa, Virginica, and each class contains 4 features, ‘Sepal length’, ‘Sepal width’, ‘Petal length’, ‘Petal width’. The aim of the iris flower classification is to predict flowers based on their specific features.

Steps to Classify Iris Flower:

1. Import libraries and Load the data
2. Analyze and visualize the dataset
3. Model training.
4. Model Evaluation.
5. Testing the model.

Import Libraries

First, we’ve imported some necessary packages for the project. Numpy will be used for any computational operations. We’ll use Matplotlib and seaborn for data visualization. Pandas help to load data from various sources like local storage, database, excel file, CSV file, etc.

In [1]:

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

Read Dataset

Next, we load the data using pd.read_csv() and set the column name as per the iris data information. Pd.read_csv reads CSV files. CSV stands for comma separated value.

In [4]:

```
df=pd.read_csv("C://Users//Abhishek//Desktop//Iris1.csv")
```

Analyze the dataset

Let’s see some information about the dataset.we will use some methods and functions to analyze the dataset.

In [5]:

```
df.head()
```

Out[5]:

	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 [6]:

```
df.tail()
```

Out[6]:

	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 [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [8]:

```
df.describe()
```

Out[8]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [9]:

```
df.columns
```

Out[9]:

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidth
Cm',
       'Species'],
      dtype='object')
```

In [10]:

```
df.mean
```

Out[10]:

```
<bound method NDFrame._add_numeric_operations.<locals>.mean of       Id  S
epalLengthCm  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 [11]:

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

Out[11]:

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

In [12]:

```
df.nunique()
```

Out[12]:

```
Id               150
SepalLengthCm     35
SepalWidthCm      23
PetalLengthCm     43
PetalWidthCm      22
Species            3
dtype: int64
```

In [14]:

```
df.count()
```

Out[14]:

```
Id               150
SepalLengthCm    150
SepalWidthCm     150
PetalLengthCm    150
PetalWidthCm     150
Species          150
dtype: int64
```

In [15]:

```
df.shape
```

Out[15]:

```
(150, 6)
```

In [16]:

```
df.size
```

Out[16]:

```
900
```

In [17]:

```
df.isnull()
```

Out[17]:

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

150 rows × 6 columns

In [18]:

```
df.mode()
```

Out[18]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.0	3.0	1.5	0.2	Iris-setosa
1	2	NaN	NaN	NaN	NaN	Iris-versicolor
2	3	NaN	NaN	NaN	NaN	Iris-virginica
3	4	NaN	NaN	NaN	NaN	NaN
4	5	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...
145	146	NaN	NaN	NaN	NaN	NaN
146	147	NaN	NaN	NaN	NaN	NaN
147	148	NaN	NaN	NaN	NaN	NaN
148	149	NaN	NaN	NaN	NaN	NaN
149	150	NaN	NaN	NaN	NaN	NaN

150 rows × 6 columns

In [19]:

```
df.median()
```

```
C:\Users\Abhishek\AppData\Local\Temp\ipykernel_376\530051474.py:1: FutureW
arning: Dropping of nuisance columns in DataFrame reductions (with 'numeri
c_only=None') is deprecated; in a future version this will raise TypeErro
r.  Select only valid columns before calling the reduction.
  df.median()
```

Out[19]:

```
Id               75.50
SepalLengthCm     5.80
SepalWidthCm      3.00
PetalLengthCm     4.35
PetalWidthCm      1.30
dtype: float64
```

In [20]:

```
df.isnull().any()
```

Out[20]:

```
Id               False
SepalLengthCm    False
SepalWidthCm     False
PetalLengthCm    False
PetalWidthCm     False
Species          False
dtype: bool
```

In [21]:

```
df.cumsum()
```

Out[21]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	3	10.0	6.5	2.8	0.4	Iris-setosaIris-setosa
2	6	14.7	9.7	4.1	0.6	Iris-setosaIris-setosaIris-setosa
3	10	19.3	12.8	5.6	0.8	Iris-setosaIris-setosaIris-setosaIris-setosa
4	15	24.3	16.4	7.0	1.0	Iris-setosaIris-setosaIris-setosaIris-setosaIr...
...	...	...	...	...	...	...
145	10731	851.6	446.2	543.1	171.8	Iris-setosaIris-setosaIris-setosaIris-setosaIr...
146	10878	857.9	448.7	548.1	173.7	Iris-setosaIris-setosaIris-setosaIris-setosaIr...
147	11026	864.4	451.7	553.3	175.7	Iris-setosaIris-setosaIris-setosaIris-setosaIr...
148	11175	870.6	455.1	558.7	178.0	Iris-setosaIris-setosaIris-setosaIris-setosaIr...
149	11325	876.5	458.1	563.8	179.8	Iris-setosaIris-setosaIris-setosaIris-setosaIr...

150 rows × 6 columns

Visualize the Dataset

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. To visualize the whole dataset we used the seaborn pair plot method. It plots the whole dataset's information.

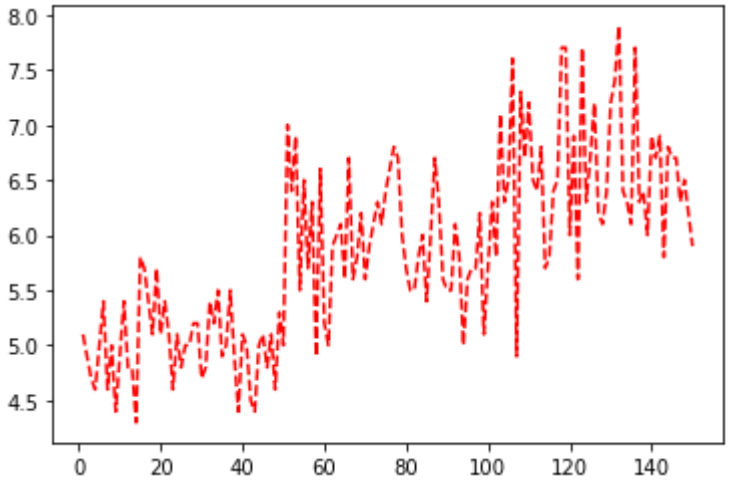
In [32]:

```
import pandas as pd
import matplotlib.pyplot as plt
iris = pd.read_csv("C://Users//Abhishek//Desktop//Iris1.csv")

plt.plot(iris.Id, iris["SepalLengthCm"], "r--")
plt.show
```

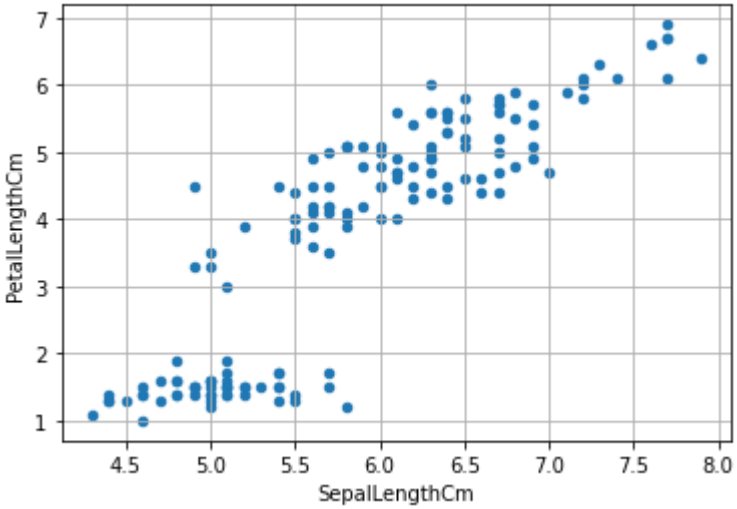
Out[32]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [34]:

```
iris.plot(kind ="scatter",
        x ='SepalLengthCm',
        y ='PetalLengthCm')
plt.grid()
```

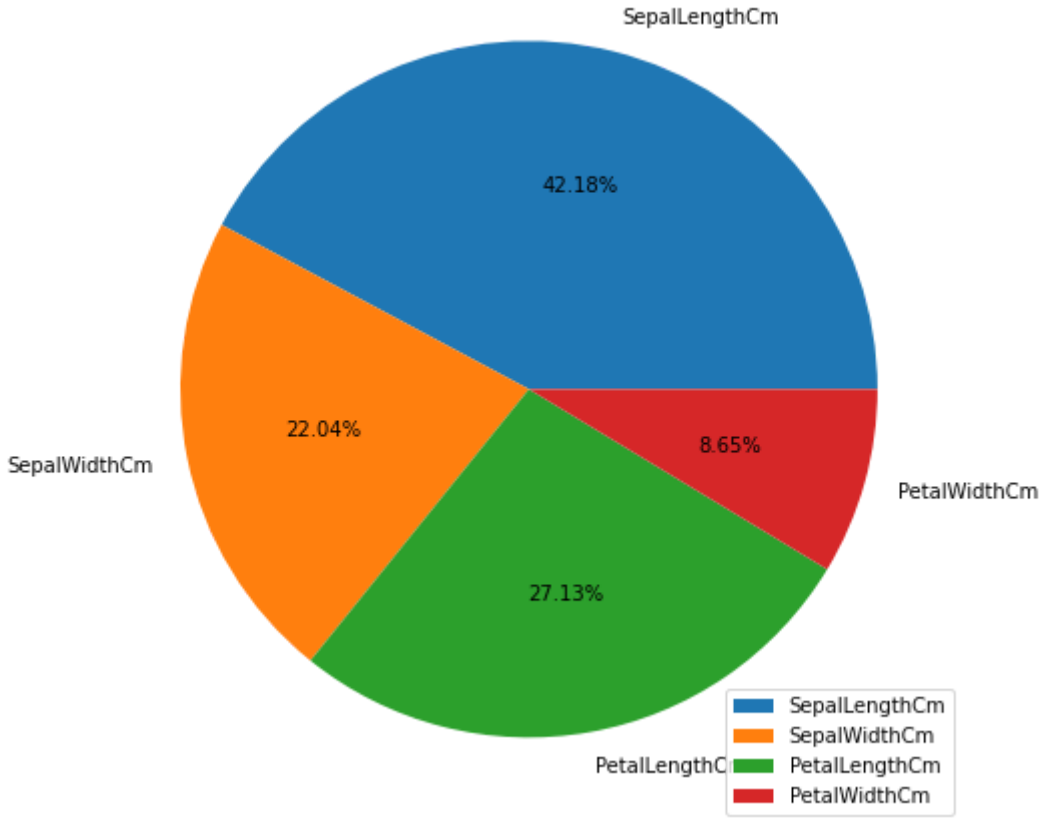


In [67]:

```
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df1=df.drop(['Id'],axis=1)
m=df1.mean()
plt.figure(figsize=(9,8))
print("SAKSHI PATEL")
plt.pie(m,labels=m.index,autopct="%.2f%%")
plt.legend(loc=4)
plt.show()
```

```
SAKSHI PATEL

C:\Users\Abhishek\AppData\Local\Temp\ipykernel_376\3885219403.py:6: Future
Warning: Dropping of nuisance columns in DataFrame reductions (with 'numer
ic_only=None') is deprecated; in a future version this will raise TypeErro
r.  Select only valid columns before calling the reduction.
  m=df1.mean()
```

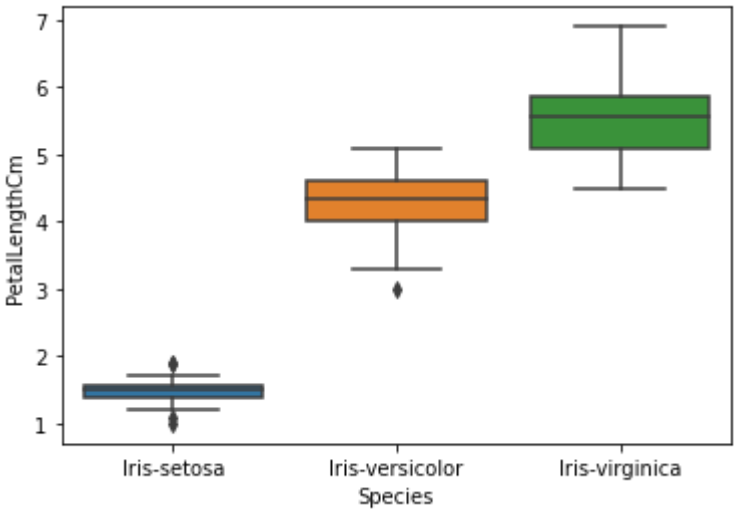


In [36]:

```
sns.boxplot(x='Species',y='PetalLengthCm',data=iris)
```

Out[36]:

```
<AxesSubplot:xlabel='Species', ylabel='PetalLengthCm'>
```



In [68]:

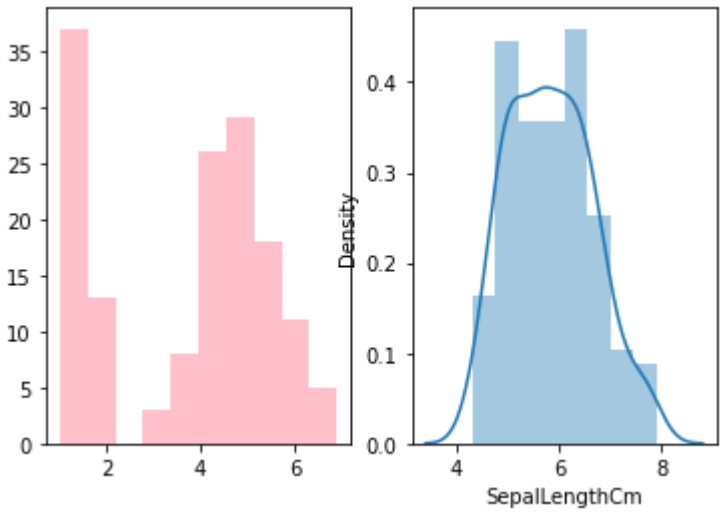
```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
print("sakshi patel")
print("20100BTCSDSI07291")
plt.subplot(1,2,1)
plt.hist(df['PetalLengthCm'].values,color='pink')
plt.subplot(1,2,2)
sns.distplot(df['SepalLengthCm'],hist=True)
plt.show
```

```
sakshi patel
20100BTCSDSI07291

C:\Users\Abhishek\anaconda3\lib\site-packages\seaborn\distributions.py:261
9: FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a fig
ure-level function with similar flexibility) or `histplot` (an axes-level
function for histograms).
  warnings.warn(msg, FutureWarning)
```

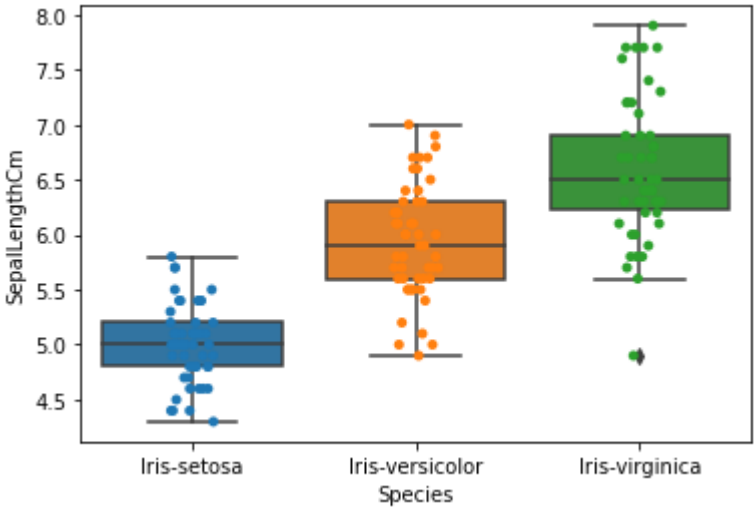
Out[68]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [38]:

```
#Combining BoxPlot and StripPlot
ax=sns.boxplot(x='Species',y='SepalLengthCm',data=iris)
ax=sns.stripplot(x='Species',y='SepalLengthCm',data=iris,jitter=True,edgecolor='gray')
```

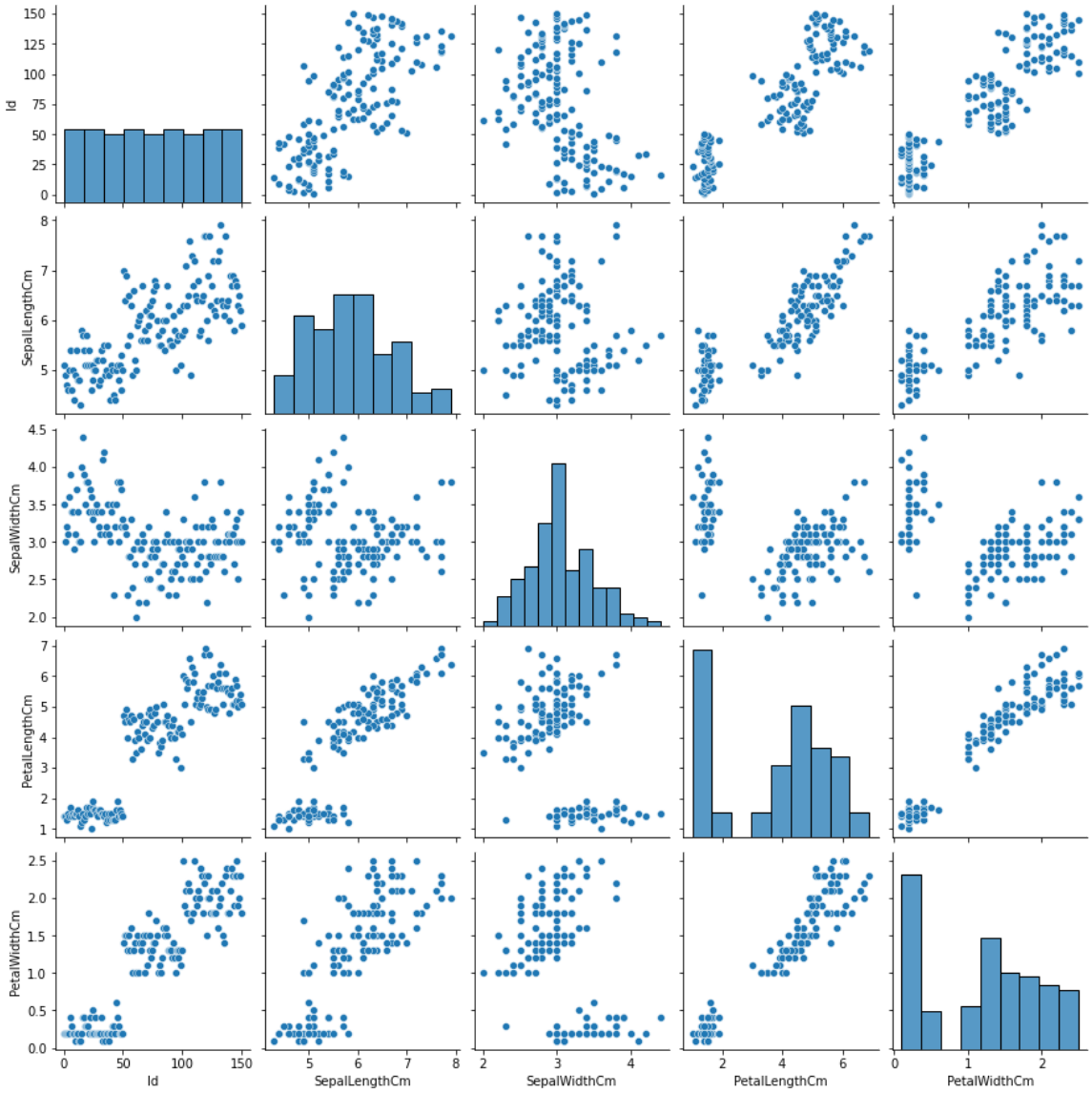


In [40]:

```
sns.pairplot(data=iris,kind='scatter')
```

Out[40]:

```
<seaborn.axisgrid.PairGrid at 0x285e91c3730>
```

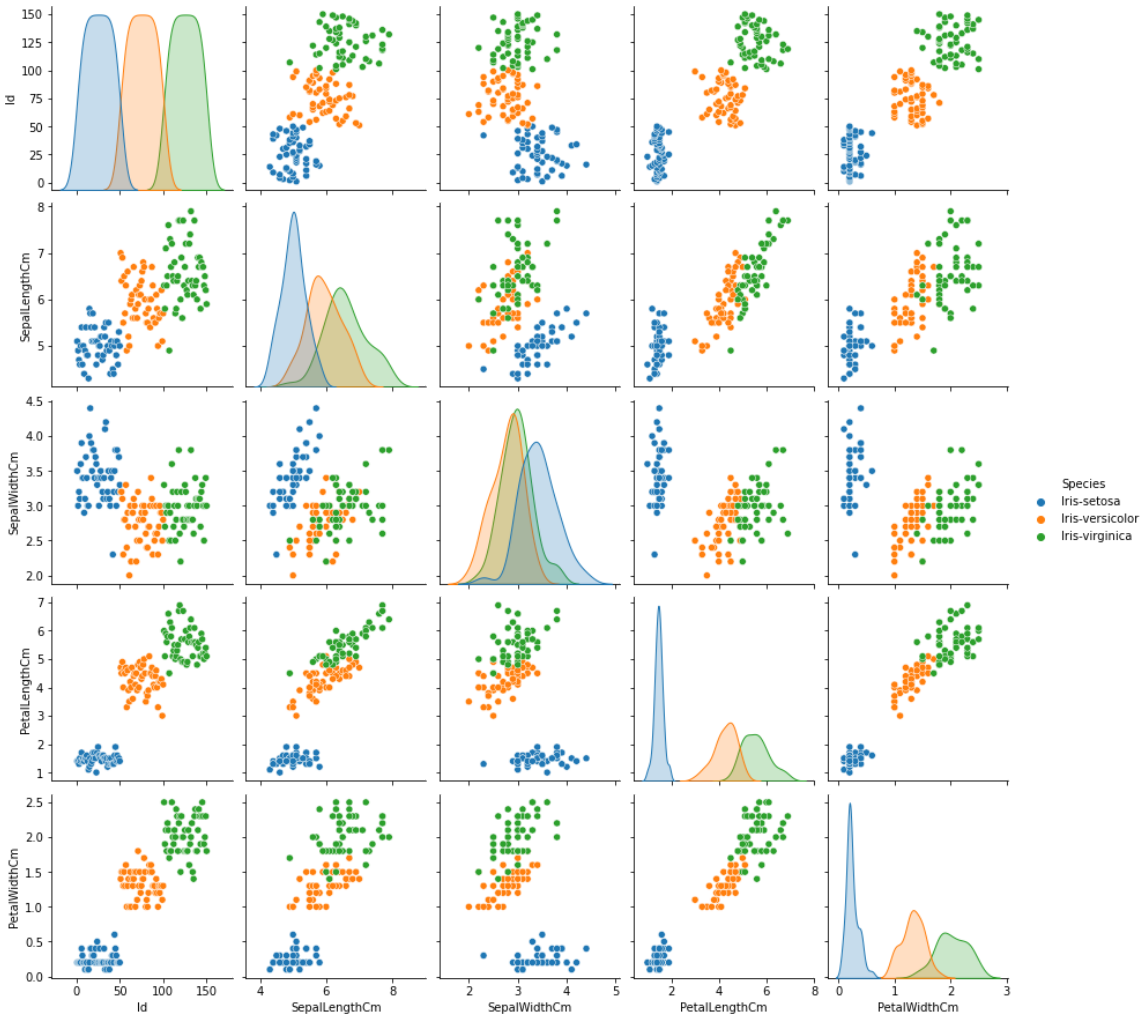


In [41]:

```
sns.pairplot(iris,hue='Species')
```

Out[41]:

```
<seaborn.axisgrid.PairGrid at 0x285e7fe01c0>
```



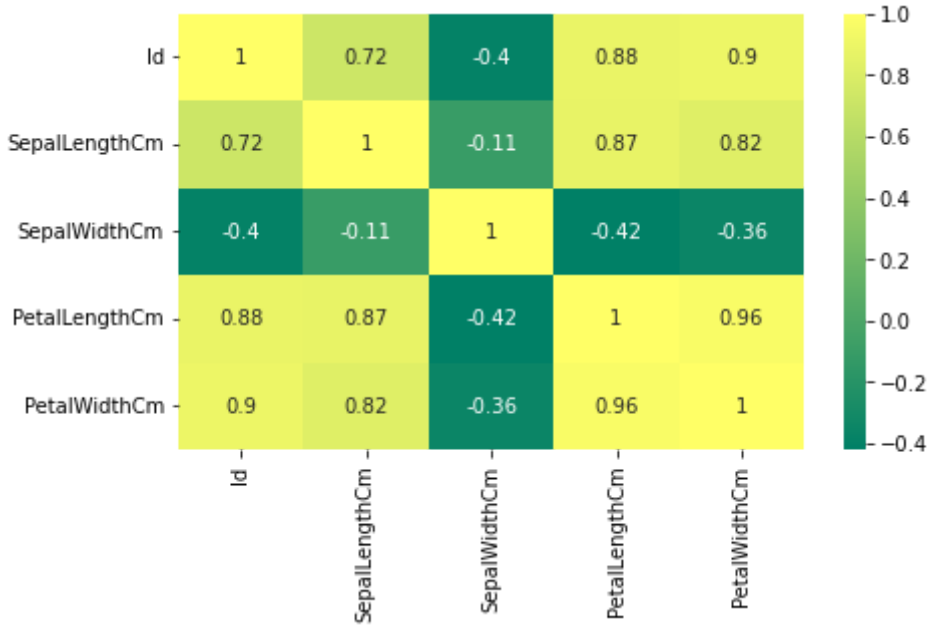
Plotting Heat Map

In [42]:

```
plt.figure(figsize=(7,4))
sns.heatmap(iris.corr(),annot=True,cmap='summer')
```

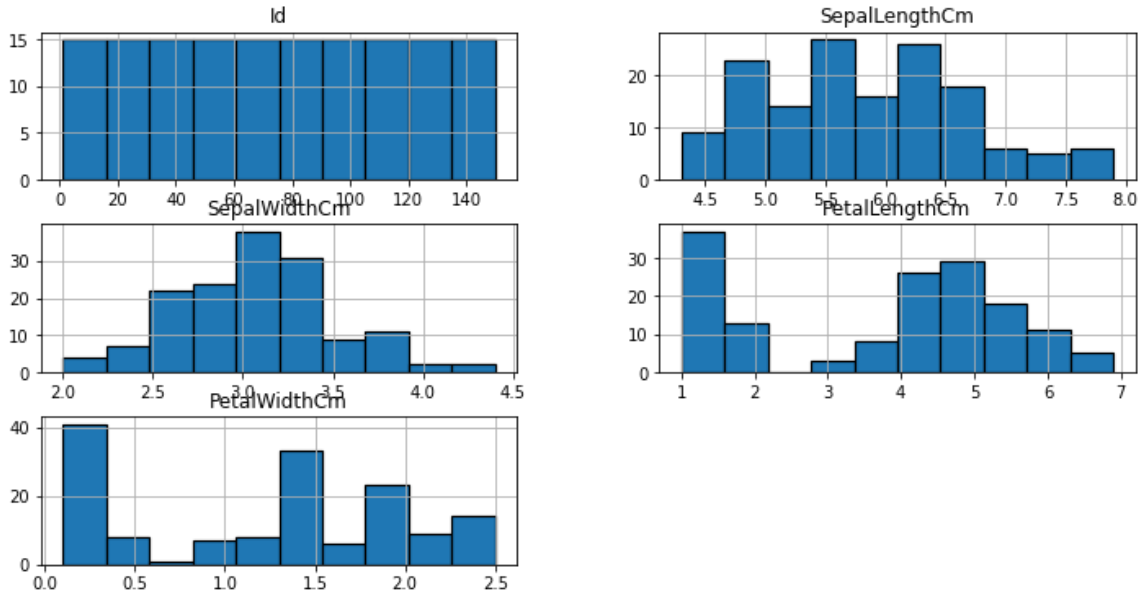
Out[42]:

<AxesSubplot:>



In [43]:

```
iris.hist(edgecolor='black', linewidth=1.2)
fig=plt.gcf()
fig.set_size_inches(12,6)
plt.show()
```



In [71]:

```
y = iris["Species"].values
y[0:5]
```

Out[71]:

```
array(['Iris-setosa', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa'], dtype=object)
```

Spliting data into Test-Train dataset

Using train_test_split we split the whole data into training and testing datasets. Later we'll use the testing dataset to check the accuracy of the model.

In [72]:

```
from sklearn.model_selection import train_test_split
X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.3, rand
X_trainset.shape, X_testset.shape, y_trainset.shape, y_testset.shape
```

Out[72]:

```
((105, 4), (45, 4), (105,), (45,))
```

Modelling and Fitting the Decision Tree Classifier

In [74]:

```
from sklearn.tree import DecisionTreeClassifier
iris_tree = DecisionTreeClassifier(criterion="entropy", max_depth = 4,random_state=1234)
iris_tree

#fiting the model
iris_tree.fit(X_trainset,y_trainset)
```

Out[74]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=123
4)
```

Prediction

In [77]:

```
predtree = iris_tree.predict(X_testset)
```

In [78]:

```
print(predtree[15:25])
print(y_testset[15:25])
```

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

Accuracy Evaluation

In [79]:

```
from sklearn import metrics
print("DecisionTrees's Accuracy: ", round(metrics.accuracy_score(y_testset, predtree),2))
```

```
DecisionTrees's Accuracy:  1.0
```