In [40]:

```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt
from sklearn import metrics
from  sklearn.model_selection import train_test_split
```

In [22]:

```
data = pd.read_csv("Iris.csv")
data.sample(5)
```

Out[22]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
142	143	5.8	2.7	5.1	1.9	Iris-virginica
71	72	6.1	2.8	4.0	1.3	Iris-versicolor
129	130	7.2	3.0	5.8	1.6	Iris-virginica
38	39	4.4	3.0	1.3	0.2	Iris-setosa
75	76	6.6	3.0	4.4	1.4	Iris-versicolor

In [23]:

```
data.head(5)
```

Out[23]:

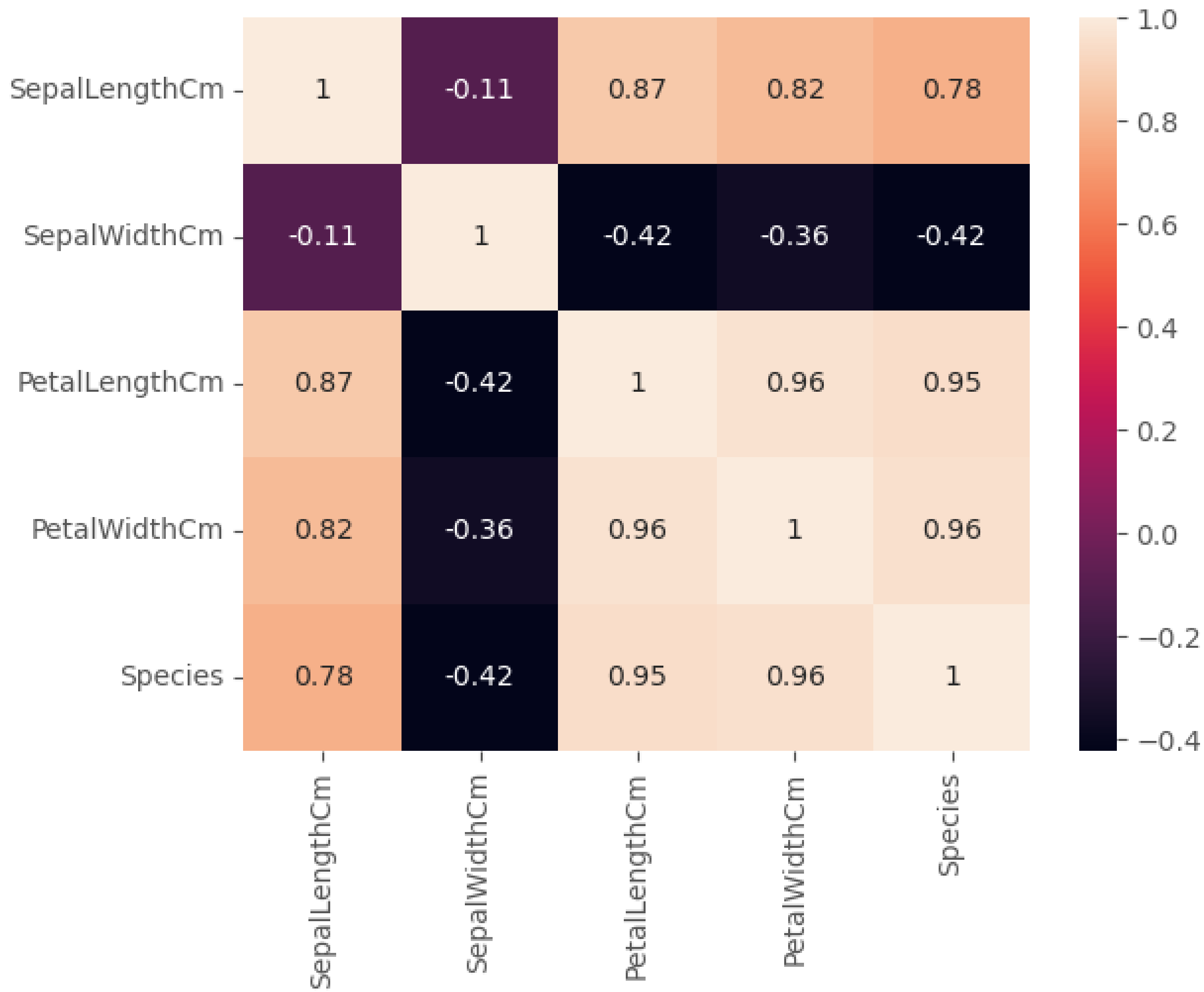
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [41]:

```
data=pd.read_csv('Iris.csv')

y=data.pop('Species')
data.pop('Id')
```

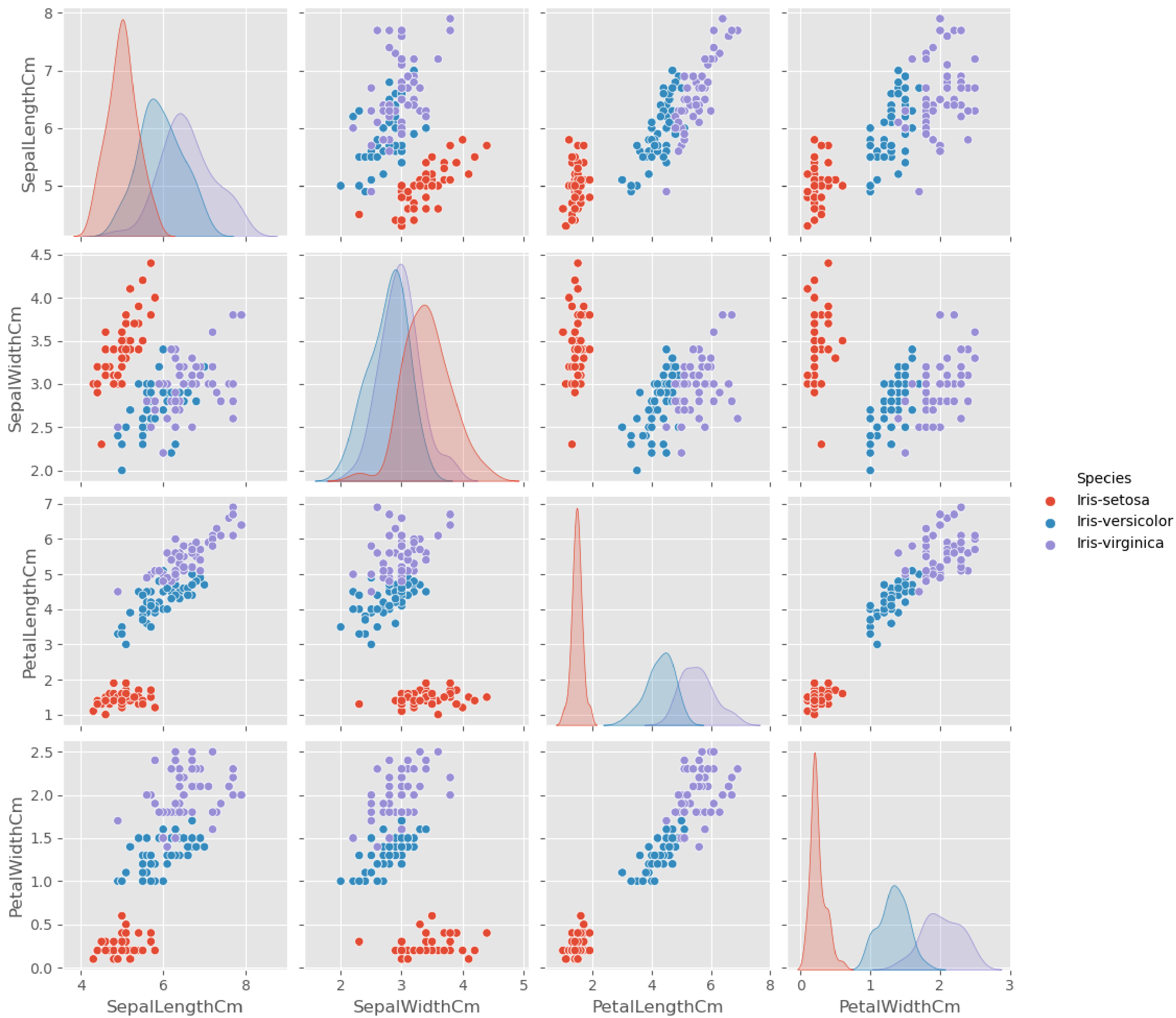
```
species=np.unique(y)

y=y.map(dict(zip(np.unique(y),np.arange(len(np.unique(y))))))
```

In [42]:
```
corr=df.corr()
sns.heatmap(corr,annot=True)
plt.show()
```



In [24]:
```
sns.pairplot( data=data, vars=('SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm'), hue='Species' )
```

Out[24]:
```
<seaborn.axisgrid.PairGrid at 0x1844dc67c70>
```


SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm
Species
Iris-setosa
Iris-versicolor
Iris-virginica

In [25]:
```
data.describe()
```

Out[25]:

	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 [26]:
```
df_norm = data[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']].apply(lambda x: (x - x.min()) / (x.m
df_norm.sample(n=5)
```

Out[26]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
147	0.611111	0.416667	0.711864	0.791667
107	0.833333	0.375000	0.898305	0.708333
99	0.388889	0.333333	0.525424	0.500000
148	0.527778	0.583333	0.745763	0.916667
75	0.638889	0.416667	0.576271	0.541667

In [28]:
```
df_norm.describe()
```

Out[28]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000
mean	0.428704	0.439167	0.467571	0.457778
std	0.230018	0.180664	0.299054	0.317984
min	0.000000	0.000000	0.000000	0.000000
25%	0.222222	0.333333	0.101695	0.083333
50%	0.416667	0.416667	0.567797	0.500000
75%	0.583333	0.541667	0.694915	0.708333
max	1.000000	1.000000	1.000000	1.000000

In [29]:

```
target = data[['Species']].replace(['Iris-setosa','Iris-versicolor','Iris-virginica'],[0,1,2])
target.sample(n=5)
```

Out[29]:

	Species
18	0
30	0
42	0
36	0
139	2

In [30]:

```
df = pd.concat([df_norm, target], axis=1)
df.sample(n=5)
```

Out[30]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
36	0.333333	0.625000	0.050847	0.041667	0
64	0.361111	0.375000	0.440678	0.500000	1
115	0.583333	0.500000	0.728814	0.916667	2
136	0.555556	0.583333	0.779661	0.958333	2
116	0.611111	0.416667	0.762712	0.708333	2

In [31]:

```
train, test = train_test_split(df, test_size = 0.3)
trainX = train[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']]# taking the training data features
trainY=train.Species# output of our training data
testX= test[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']] # taking test data features
testY =test.Species   #output value of test data
trainX.head(5)
```

Out[31]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
134	0.500000	0.250000	0.779661	0.541667
47	0.083333	0.500000	0.067797	0.041667
118	0.944444	0.250000	1.000000	0.916667
82	0.416667	0.291667	0.491525	0.458333
94	0.361111	0.291667	0.542373	0.500000

In [32]:

```
trainY.head(5)
```

Out[32]:

```
134    2
47     0
118    2
82     1
94     1
Name: Species, dtype: int64
```

In [33]:

```
testX.head(5)
```

Out[33]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
17	0.222222	0.625000	0.067797	0.083333
137	0.583333	0.458333	0.762712	0.708333
20	0.305556	0.583333	0.118644	0.041667
6	0.083333	0.583333	0.067797	0.083333
93	0.194444	0.125000	0.389831	0.375000

In [34]:

```
testY.head(5)
```

Out[34]:

```
17     0
137    2
20     0
6      0
93     1
Name: Species, dtype: int64
```

In [35]:

```
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(3, 3), random_state=1)
```

In [36]:

```
clf.fit(trainX, trainY)
```

Out[36]:

```
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(3, 3), random_state=1,
              solver='lbfgs')
```

In [37]:

```
prediction = clf.predict(testX)
print(prediction)
```

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

In [38]:

```
print(testY.values)
```

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

In [39]:

```
print('The accuracy of the Multi-layer Perceptron is:',metrics.accuracy_score(prediction,testY))
```

```
The accuracy of the Multi-layer Perceptron is: 0.9333333333333333
```