Mini_Project_2-On-Iris-Dataset

Isaiah Melvin Issac (183CS21014)

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

In [2]:
```
df= pd.read_csv("C:/Users/Dell/Downloads/Iris.csv")
df.head()
```

Out[2]:

	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 [3]:
```
df.describe()
```

Out[3]:

	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 [4]:
```
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 [5]:
```
df.shape
```

Out[5]:
```
(150, 6)
```

In [6]:
```
df.drop("Id",axis=1,inplace=True)
df.head()
```

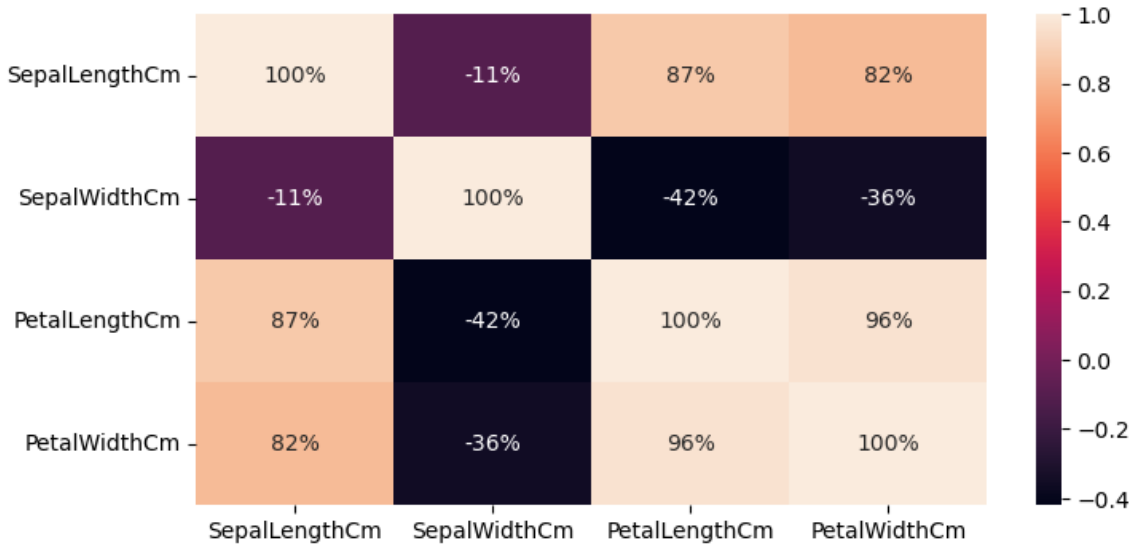
Out[6]:

	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 [ ]:
```
```

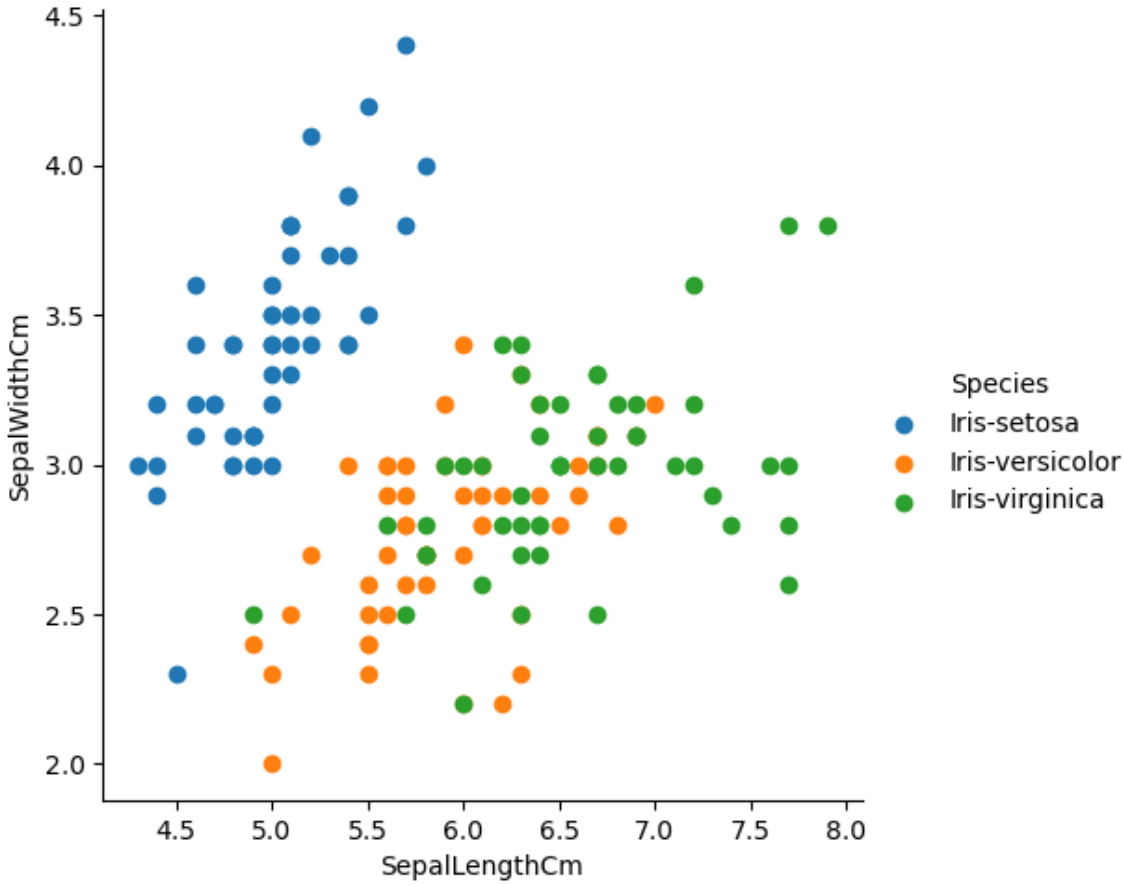
In [23]:
```
plt.figure(figsize=(8,4))
sns.heatmap(df.corr(),annot=True,fmt=".0%") #draws  heatmap with input as th
plt.show()
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_2112\901673635.py:2: FutureWarni
ng: The default value of numeric_only in DataFrame.corr is deprecated. In a
future version, it will default to False. Select only valid columns or spec
ify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(),annot=True,fmt=".0%") #draws  heatmap with input as
the correlation matrix calculted by(df.corr())
```



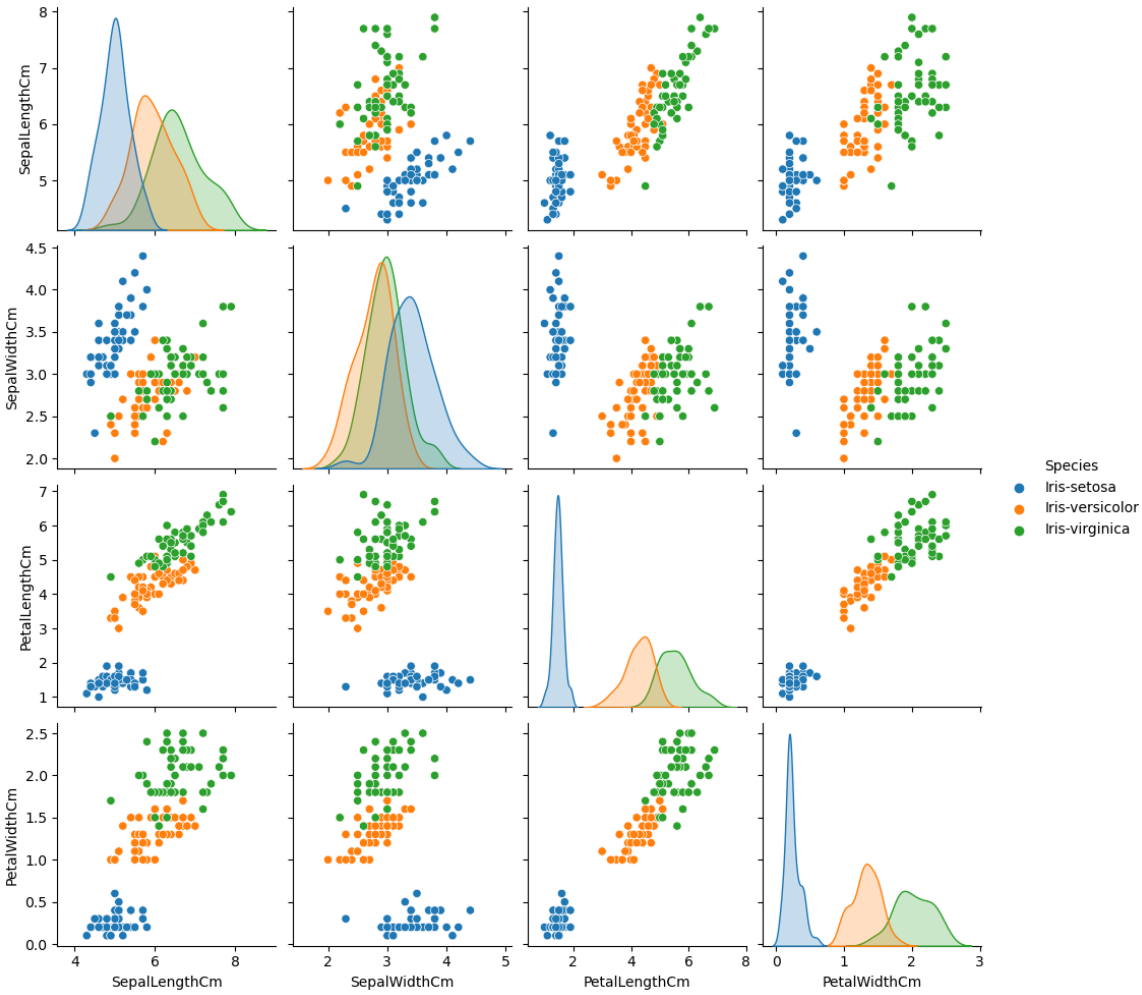
In [9]:
```
sns.FacetGrid(df, hue="Species", height=5).map(plt.scatter, "SepalLengthCm",
```

Out[9]:
```
<seaborn.axisgrid.FacetGrid at 0x1f6de85da10>
```



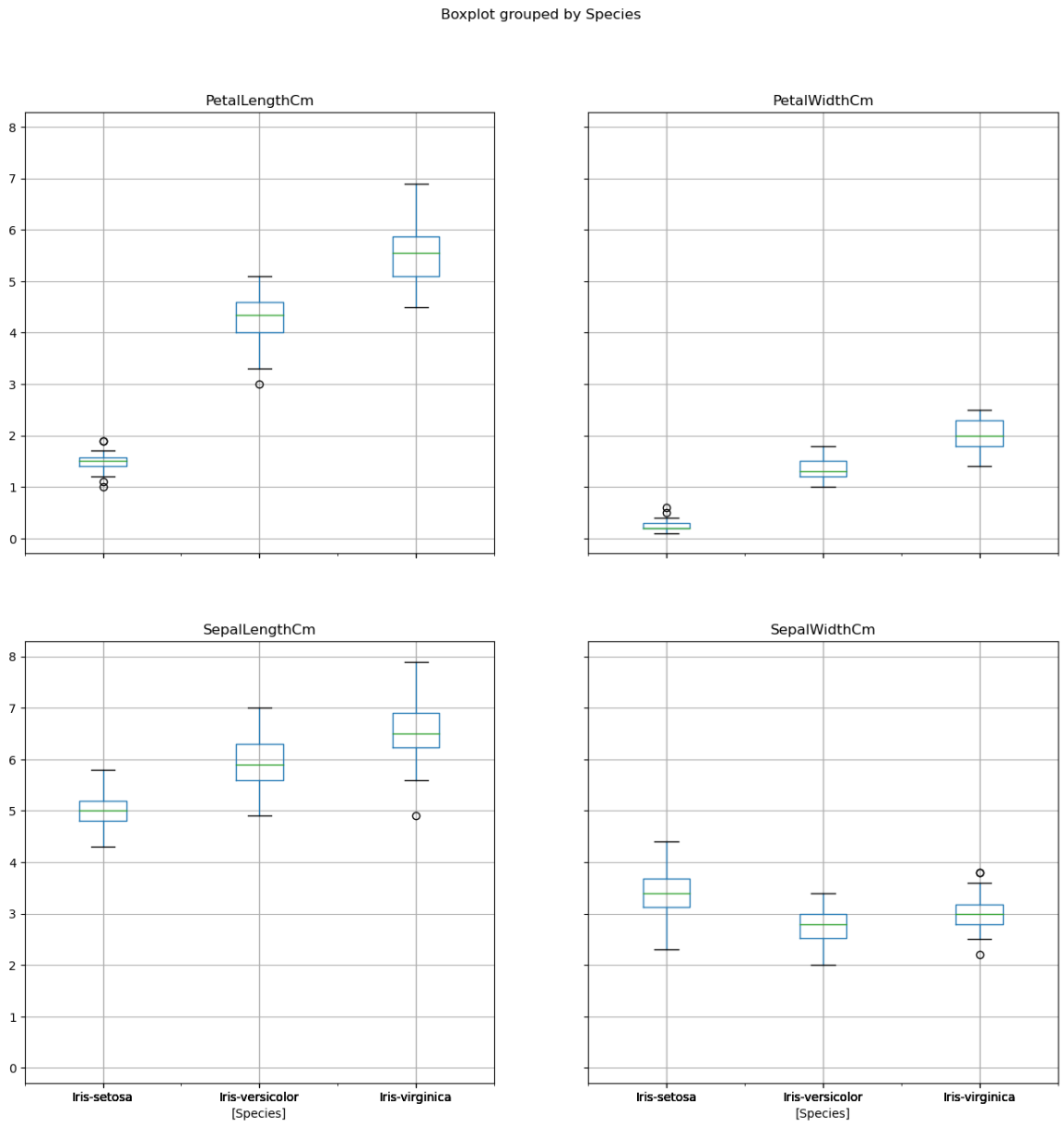
In [10]:
```
sns.pairplot(df.iloc[:,:],hue='Species')
```

Out[10]: <seaborn.axisgrid.PairGrid at 0x1f6de7d3690>



In [11]:
```
df.boxplot(by="Species", figsize=(15,15))
```

Out[11]:
```
array([[<Axes: title={'center': 'PetalLengthCm'}, xlabel='[Species]'>,
        <Axes: title={'center': 'PetalWidthCm'}, xlabel='[Species]'>],
       [<Axes: title={'center': 'SepalLengthCm'}, xlabel='[Species]'>,
        <Axes: title={'center': 'SepalWidthCm'}, xlabel='[Species]'>]],
      dtype=object)
```



In [12]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

In [13]:
```
X=df.iloc[:,0:4]
Y=df["Species"]
X.head()
```

Out[13]:

	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

In [14]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25,random_sta
print("Train Shape",X_train.shape)
print("Test Shape",X_test.shape)
```

```
Train Shape (112, 4)
Test Shape (38, 4)
```

In [15]:
```
#LOGISTIC REGRESSION
log = LogisticRegression()
log.fit(X_train,Y_train)
prediction=log.predict(X_test)
print('The accuracy of the Logistic Regression is',metrics.accuracy_score(pr
```

```
The accuracy of the Logistic Regression is 0.9736842105263158
```

In [ ]:
```

```

In [16]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

In [17]:
```
#DECISION TREE CLASSIFIER
tree=DecisionTreeClassifier()
tree.fit(X_train,Y_train)
prediction=tree.predict(X_test)
print('The accuracy of the Decision Tree is',metrics.accuracy_score(predicti
```

```
The accuracy of the Decision Tree is 0.9736842105263158
```

In [18]:
```
#K-NEIGHBOUR CLASSIFIER
knn=KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train,Y_train)
prediction=knn.predict(X_test)
print('The accuracy of the KNN is',metrics.accuracy_score(prediction,Y_test)
```

```
The accuracy of the KNN is 0.9736842105263158
```

In [19]:
```
#SUPPORT VECTOR CLASSIFIER
svc=SVC()
svc.fit(X_train,Y_train)
prediction=svc.predict(X_test)
print('The accuracy of the SVC is',metrics.accuracy_score(prediction,Y_test)
```

```
The accuracy of the SVC is 0.9736842105263158
```

In [20]:
```
#RANDOM FOREST CLASSIFIER
from sklearn.ensemble import RandomForestClassifier
forest=RandomForestClassifier(n_estimators=10,criterion="entropy",random_sta
forest.fit(X_train,Y_train)
print('The accuracy of the SVC is',metrics.accuracy_score(prediction,Y_test)
```

```
The accuracy of the SVC is 0.9736842105263158
```

In [ ]: