Introduction

The Iris dataset is a classic dataset in the field of machine learning and statistics. It consists of 150 samples of iris flowers, each belonging to one of three species: Setosa, Versicolor, and Virginica. Each sample includes four features: sepal length, sepal width, petal length, and petal width, all measured in centimeters.

Importing Necessary libraries

In [1]:
```
import pandas as pd
from sklearn.datasets import load_iris, load_diabetes
from sklearn.preprocessing import StandardScaler, LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading IRIS dataset

In [2]:
```
file_path=r"C:\Users\Irene Chelsia\Downloads\Iris.csv"
df=pd.read_csv(file_path)
df
```

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
...	...	...	...	...	...	...
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

150 rows × 6 columns

Data Understanding

In [3]:
```
df.head()
```
Out[3]:

	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 [4]:
```
df.tail()
```
Out[4]:

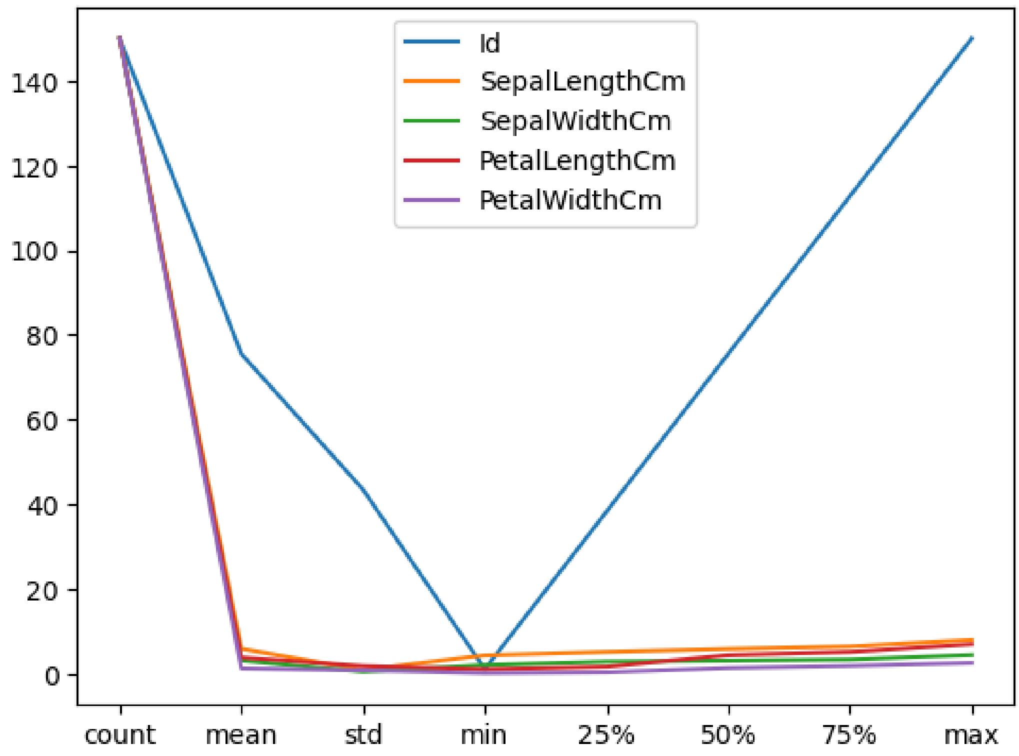
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [5]:
```
df.describe()
```
Out[5]:

	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 [6]:
```
#Statistical description of the data
df.describe().plot()
```

Out[6]: <Axes: >



In [7]:
```
df.info()
```

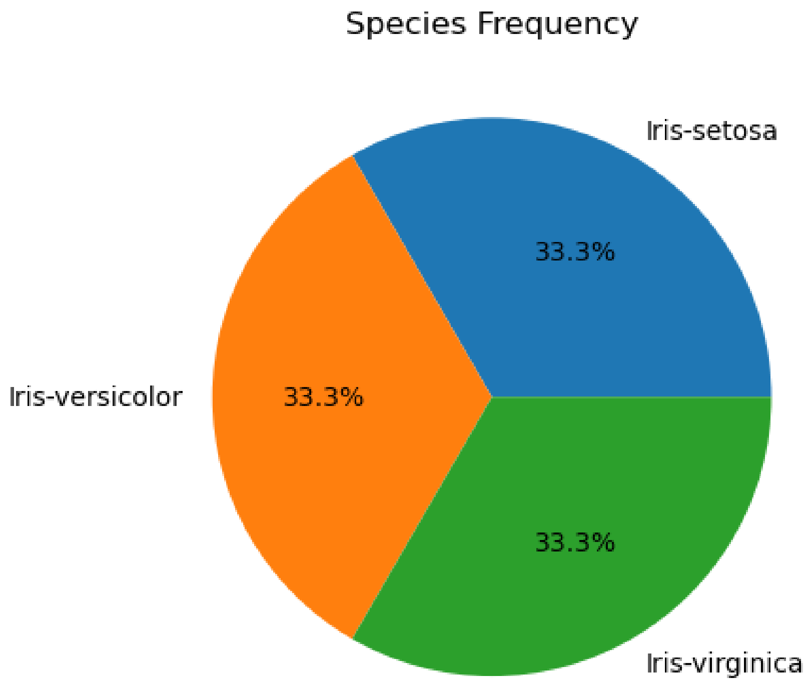
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Id             150 non-null    int64  
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   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.dtypes
```

Out[8]:
```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

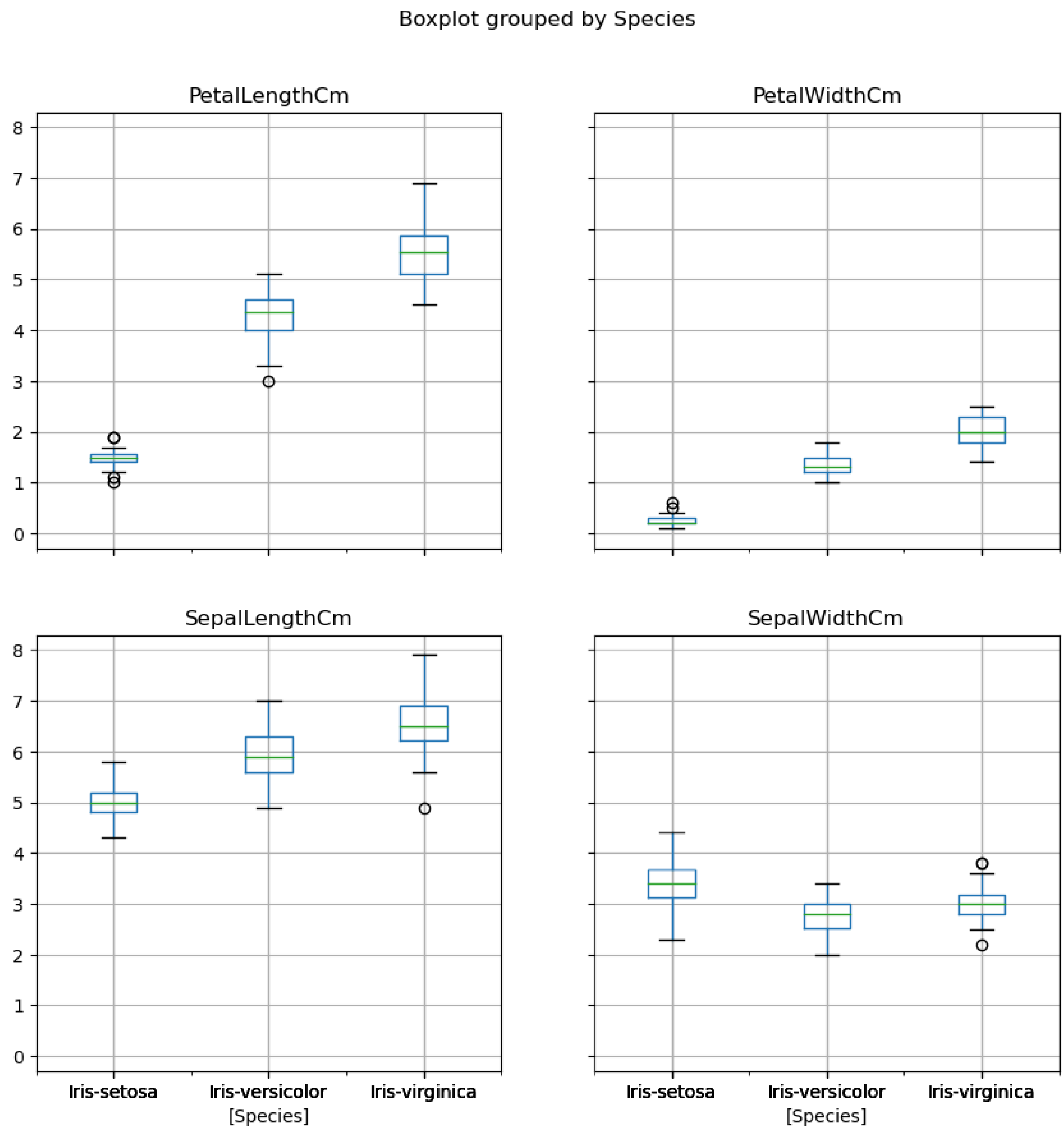
In [10]:
```
df.shape
```
Out[10]:
```
(150, 6)
```
In [11]:
```
len(df)
```
Out[11]:
```
150
```
In [12]:
```
df['Species'].unique()
d2=df['Species'].value_counts()
plt.pie(d2,labels=['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],autopct='%1.
plt.title('Species Frequency')
plt.show()
```


In [13]:
```
df['Species'].nunique()
```
Out[13]:
```
3
```
In [14]:
```
df.duplicated().sum()
```
Out[14]:
```
0
```
In [15]:
```
df[df.duplicated()]
```
Out[15]:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species

In [16]:
```
df.isnull().sum()
```
Out[16]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [17]:
```
df.drop("Id", axis=1).boxplot(by="Species", figsize=(10, 10))
plt.show()
```



In [18]:
```
df.corr()
```

```
C:\Users\Irene Chelsia\AppData\Local\Temp\ipykernel_13692\1134722465.py:1: FutureW
arning: The default value of numeric_only in DataFrame.corr is deprecated. In a fu
ture version, it will default to False. Select only valid columns or specify the v
alue of numeric_only to silence this warning.
  df.corr()
```

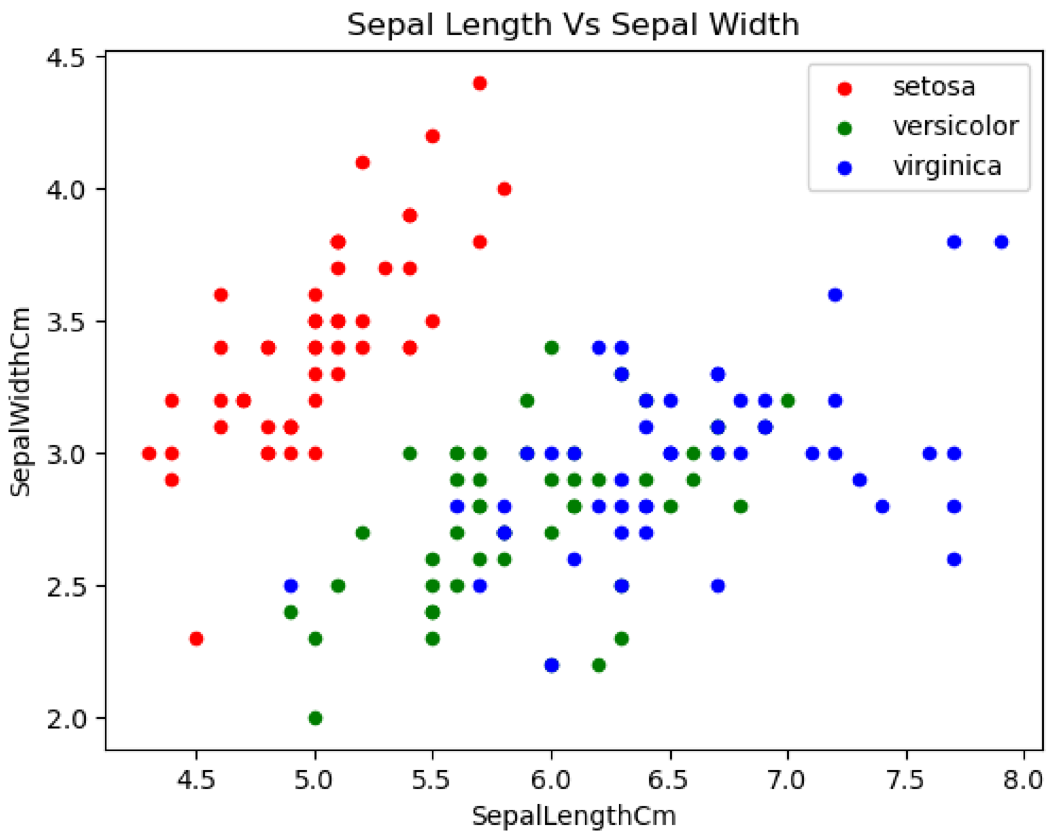
Out[18]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.716676	-0.397729	0.882747	0.899759
SepalLengthCm	0.716676	1.000000	-0.109369	0.871754	0.817954
SepalWidthCm	-0.397729	-0.109369	1.000000	-0.420516	-0.356544
PetalLengthCm	0.882747	0.871754	-0.420516	1.000000	0.962757
PetalWidthCm	0.899759	0.817954	-0.356544	0.962757	1.000000

In [19]:
```python
iris=pd.read_csv(file_path)

ax = iris[iris.Species=='Iris-setosa'].plot.scatter(x='SepalLengthCm', y='SepalWidt
                                                    color='red', label='setosa')
iris[iris.Species=='Iris-versicolor'].plot.scatter(x='SepalLengthCm', y='SepalWidth
                                                color='green', label='versicolor', 
iris[iris.Species=='Iris-virginica'].plot.scatter(x='SepalLengthCm', y='SepalWidthC
                                                color='blue', label='virginica', ax
ax.set_title("Sepal Length Vs Sepal Width")
```

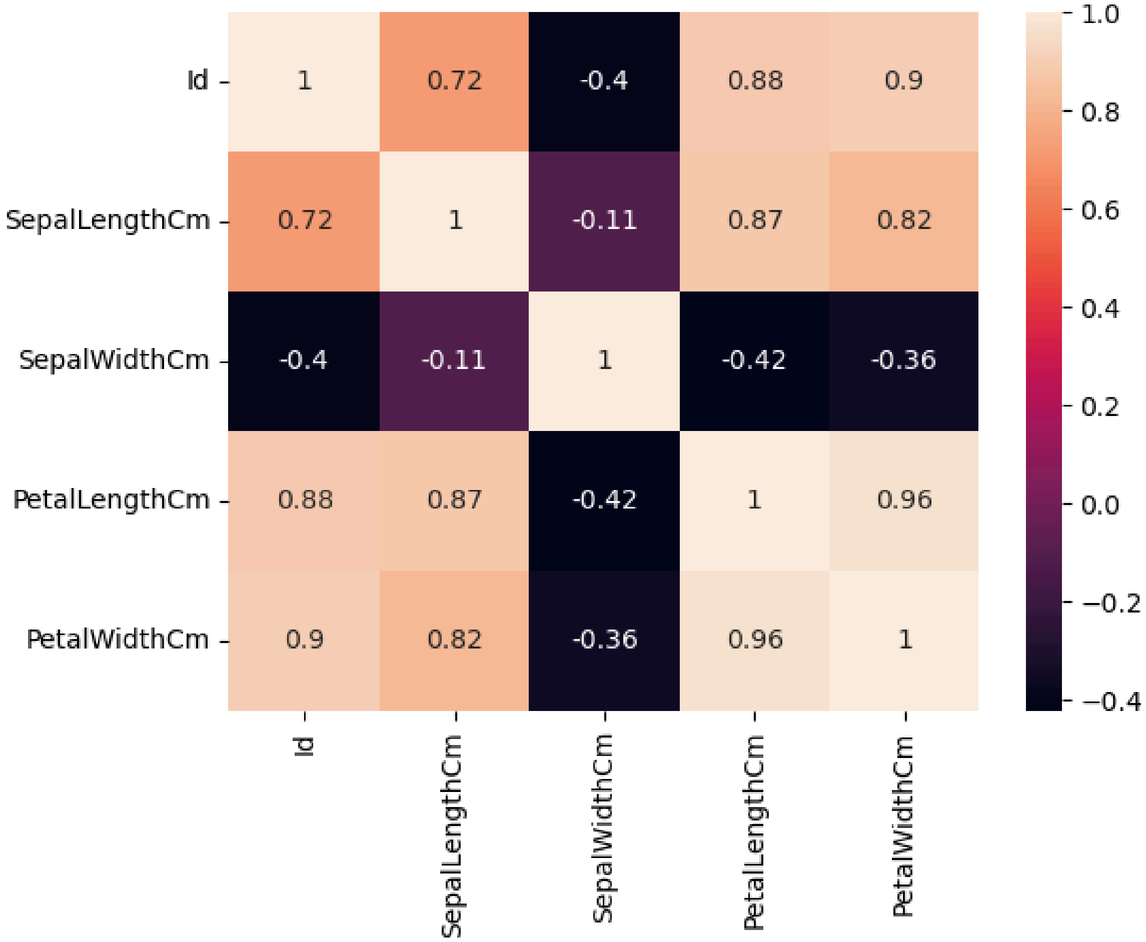
Out[19]: Text(0.5, 1.0, 'Sepal Length Vs Sepal Width')



In [20]:
```
corr=df.corr()
sns.heatmap(corr,annot=True)
```

```
C:\Users\Irene Chelsia\AppData\Local\Temp\ipykernel_13692\2699745944.py:1: FutureW
arning: The default value of numeric_only in DataFrame.corr is deprecated. In a fu
ture version, it will default to False. Select only valid columns or specify the v
alue of numeric_only to silence this warning.
  corr=df.corr()
```

Out[20]: <Axes: >



Data Preprocessing

In [21]:
```
df.drop(columns=['Id'],inplace=True)
df.head()
```

Out[21]:

	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 [22]:
```
df[['Prefix','Species']]=df['Species'].str.split('-',1,expand=True)
```

```
C:\Users\Irene Chelsia\AppData\Local\Temp\ipykernel_13692\1756609110.py:1: FutureW
arning: In a future version of pandas all arguments of StringMethods.split except
for the argument 'pat' will be keyword-only.
  df[['Prefix','Species']]=df['Species'].str.split('-',1,expand=True)
```

In [23]:
```
df.drop('Prefix',axis=1,inplace=True)
```

In [24]:
```
df['Sepal ratio']=df['SepalLengthCm']/df['SepalWidthCm']
df['Petal ratio']=df['PetalLengthCm']/df['PetalWidthCm']
```

In [25]:
```
df.head()
```

Out[25]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	Sepal ratio	Petal ratio
0	5.1	3.5	1.4	0.2	setosa	1.457143	7.0
1	4.9	3.0	1.4	0.2	setosa	1.633333	7.0
2	4.7	3.2	1.3	0.2	setosa	1.468750	6.5
3	4.6	3.1	1.5	0.2	setosa	1.483871	7.5
4	5.0	3.6	1.4	0.2	setosa	1.388889	7.0

Feature Engineering

In [26]:
```
minmax=MinMaxScaler()
df[['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm','Sepal ratio','Pe

df.head()
```

Out[26]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	Sepal ratio	Petal ratio
0	0.222222	0.625000	0.067797	0.041667	setosa	0.111531	0.378641
1	0.166667	0.416667	0.067797	0.041667	setosa	0.215586	0.378641
2	0.111111	0.500000	0.050847	0.041667	setosa	0.118386	0.339806
3	0.083333	0.458333	0.084746	0.041667	setosa	0.127317	0.417476
4	0.194444	0.666667	0.067797	0.041667	setosa	0.071222	0.378641

In [27]:
```
encoder=LabelEncoder()
df['Species']=encoder.fit_transform(df['Species'])

df.head()
```

Out[27]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	Sepal ratio	Petal ratio
0	0.222222	0.625000	0.067797	0.041667	0	0.111531	0.378641
1	0.166667	0.416667	0.067797	0.041667	0	0.215586	0.378641
2	0.111111	0.500000	0.050847	0.041667	0	0.118386	0.339806
3	0.083333	0.458333	0.084746	0.041667	0	0.127317	0.417476
4	0.194444	0.666667	0.067797	0.041667	0	0.071222	0.378641

In [28]:
```
labels = ['Small', 'Medium', 'Tall']
df['Sepal bin'] = pd.cut(df['Sepal ratio'], bins=3, labels=labels)
df['Petal bin'] = pd.cut(df['Petal ratio'], bins=3, labels=labels)

df.head()
```

Out[28]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	Sepal ratio	Petal ratio	Sepal bin
0	0.222222	0.625000	0.067797	0.041667	0	0.111531	0.378641	Small
1	0.166667	0.416667	0.067797	0.041667	0	0.215586	0.378641	Small
2	0.111111	0.500000	0.050847	0.041667	0	0.118386	0.339806	Small
3	0.083333	0.458333	0.084746	0.041667	0	0.127317	0.417476	Small
4	0.194444	0.666667	0.067797	0.041667	0	0.071222	0.378641	Small

Split the data into training and testing sets

In [29]:
```
X=df.iloc[:,:4]
y=df.iloc[:,4:5]
```

In [30]:
```
X
```

Out[30]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	0.222222	0.625000	0.067797	0.041667
1	0.166667	0.416667	0.067797	0.041667
2	0.111111	0.500000	0.050847	0.041667
3	0.083333	0.458333	0.084746	0.041667
4	0.194444	0.666667	0.067797	0.041667
...	...	...	...	...
145	0.666667	0.416667	0.711864	0.916667
146	0.555556	0.208333	0.677966	0.750000
147	0.611111	0.416667	0.711864	0.791667
148	0.527778	0.583333	0.745763	0.916667
149	0.444444	0.416667	0.694915	0.708333

150 rows × 4 columns

In [31]:
```
y
```

Out[31]:

	Species
0	0
1	0
2	0
3	0
4	0
...	...
145	2
146	2
147	2
148	2
149	2

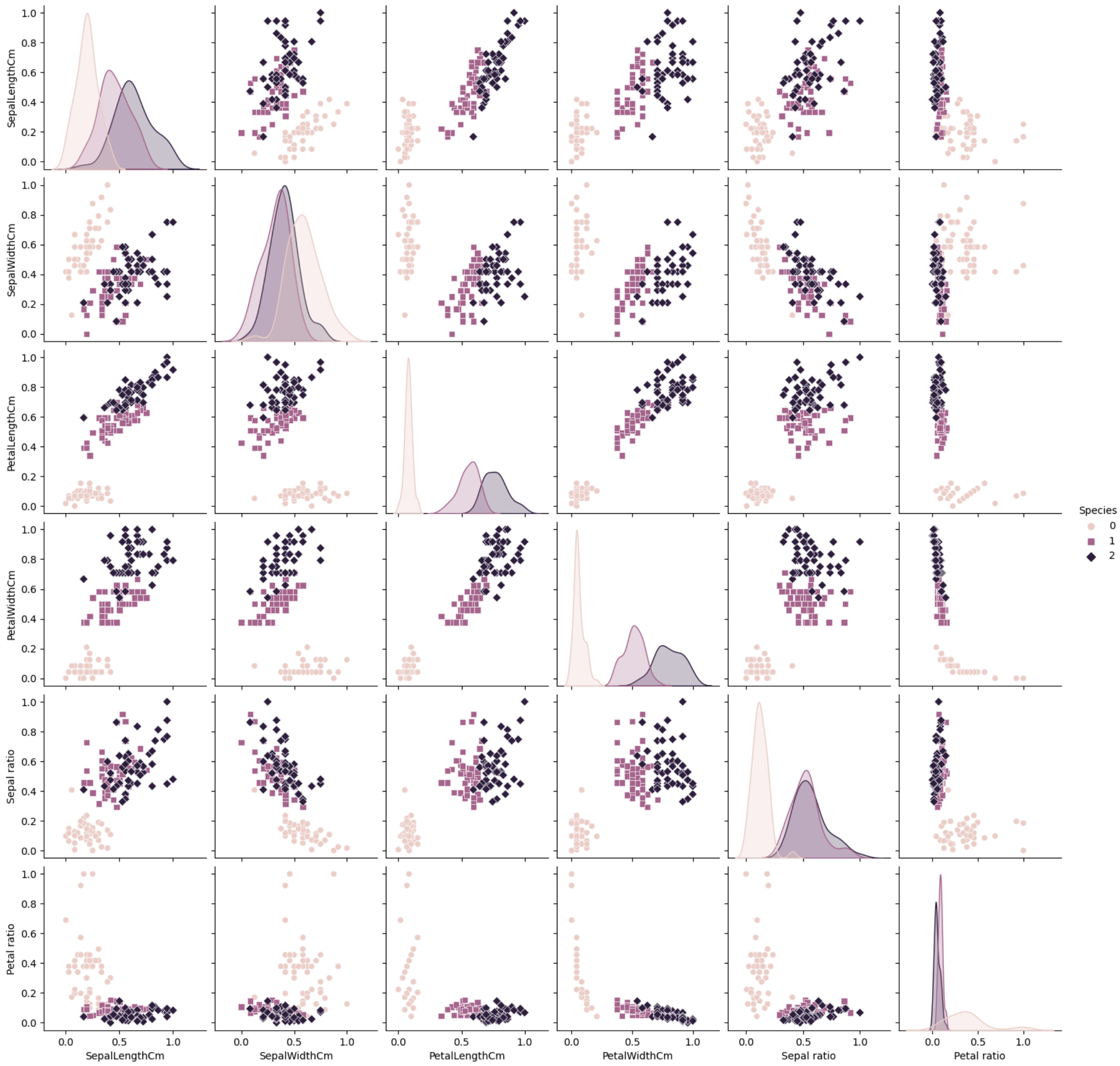
150 rows × 1 columns

In [32]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

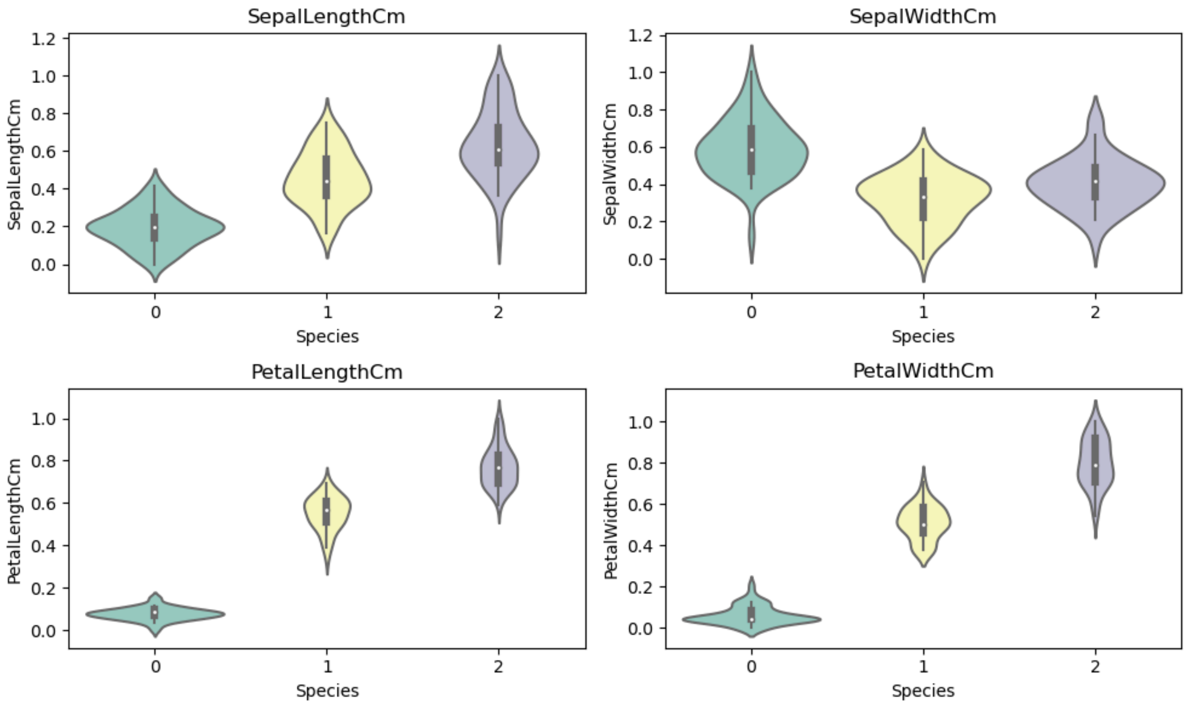
Visualization

In [33]:
```
sns.pairplot(df, hue='Species', markers=['o', 's', 'D'])
```

Out[33]:
```
<seaborn.axisgrid.PairGrid at 0x1925dd3dbd0>
```



In [34]:
```
plt.figure(figsize=(10, 6))
for i, feature in enumerate(X.columns):
    plt.subplot(2, 2, i+1)
    sns.violinplot(x='Species', y=feature, data=df, palette='Set3')
    plt.title(feature)
plt.tight_layout()
```



Random Forest Classifier

In [35]:
```
rfc=RandomForestClassifier(n_estimators=100)
rfc.fit(X_train,y_train)
```

```
C:\Users\Irene Chelsia\anacondanew1\Lib\site-packages\sklearn\base.py:1151: DataCo
nversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

Out[35]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [36]:
```
y_pred=rfc.predict(X_test)
```

Model Performance

In [37]:
```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

In [38]:
```
precision = precision_score(y_test, y_pred, average='weighted')
print("Precision:", precision)
```

```
Precision: 1.0
```

In [39]:
```
recall = recall_score(y_test, y_pred, average='weighted')
print("Recall:", recall)
```

```
Recall: 1.0
```

In [40]:
```
f1 = f1_score(y_test, y_pred, average='weighted')
print("F1 Score:", f1)
```

```
F1 Score: 1.0
```

In [41]:
```
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

In [42]:
```
roc_auc = roc_auc_score(y_test, rfc.predict_proba(X_test), multi_class='ovr')
print("ROC AUC Score:", roc_auc)
```

```
ROC AUC Score: 1.0
```

Insights

```
Species Differentiation: The dataset helps distinguish between three types of iris
flowers: Setosa, Versicolor, and Virginica, based on their petal and sepal
measurements.

Feature Importance: It shows which characteristics (sepal length, sepal width,
petal length, petal width) are most relevant for classifying iris species.

Model Selection: It aids in choosing the right machine learning model (like
decision trees or support vector machines) for accurately predicting iris species.

Model Evaluation: It helps assess how well the chosen model performs in predicting
iris species using metrics like accuracy or confusion matrix.

Real-World Applications: Insights from the dataset can be applied to various
fields such as botany or agriculture for species identification and
classification.
```