Exploratory Data Analysis on "Iris Dataset"

Project Summary The Iris dataset serves as a springboard for many machine learning projects, particularly those focused on classification. This project explores the characteristics of the Iris flower and utilizes its measurements to build a classification model.

Dataset:
Source: The dataset contains 150 samples from three Iris species: Iris setosa, Iris virginica, and Iris versicolor.
Purpose: Introduced by Ronald Fisher to showcase linear discriminant analysis
Features: Each flower is described by four features: sepal length, sepal width, petal length, and petal width (all in centimeters).
Classes: Three Iris species.

In [ ]:
```
# Importing the necessary Libraries
```

In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
```

Load Iris Dataset

In [5]:
```
data=pd.read_csv("D:/Prediction/Iris.csv")
data
```

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
...	...	...	...	...	...	...
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 Exploration and Data Cleaning

In [12]:
```
#checking what are the variables here:
data.columns
```

Out[12]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidt
hCm',
       'Species'],
      dtype='object')
```

In [5]:
```
#checking shape of Iris dataset
data.shape
```

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

In [8]:
```
#basic information about the dataset
data.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 [9]:
```
# checking null values of each columns
data.isnull().sum()
```

Out[9]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [11]:
```
data.isna().sum()
```

Out[11]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

In [9]:
```
# describe the DataFrame
data.describe()
```

Out[9]:

	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

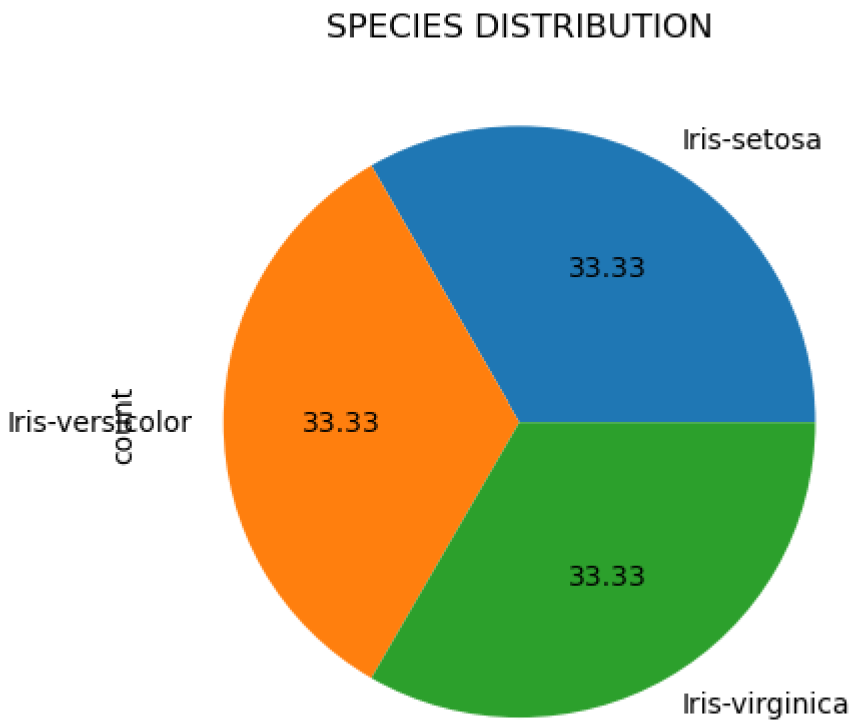
In [10]:
```
data["Species"].value_counts()
```

Out[10]:
```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: count, dtype: int64
```

Data Visualization

In [18]:
```
data["Species"].value_counts().plot(kind="pie",autopct="%.2f")
plt.title("SPECIES DISTRIBUTION")
```

Out[18]:
```
Text(0.5, 1.0, 'SPECIES DISTRIBUTION')
```



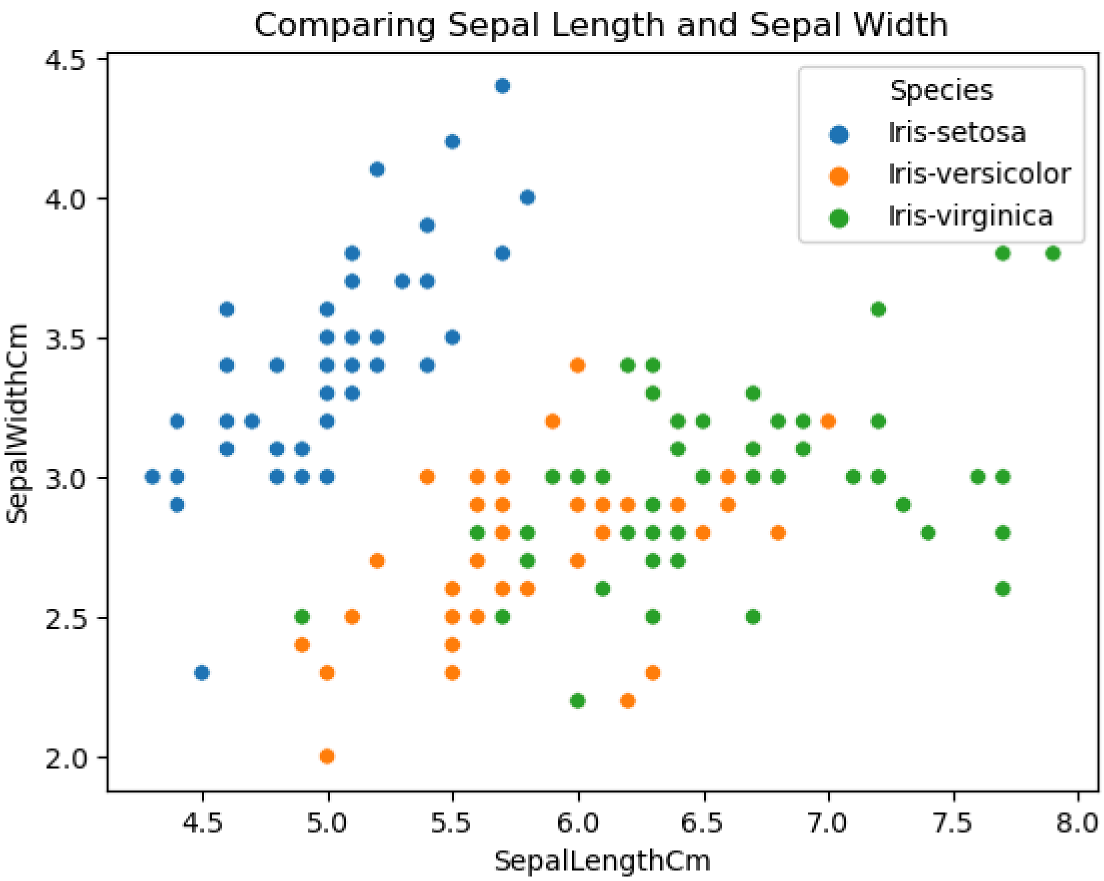
```
Data Insight:
1. This further visualizes that species are well balanced
2. Each species ( Iris virginica, setosa, versicolor) has 50 as it’s
count
```

In [13]:

```
sns.scatterplot(x='SepalLengthCm',y='SepalWidthCm',hue='Species',data=data)
plt.title("Comparing Sepal Length and Sepal Width")
```

Out[13]:

```
Text(0.5, 1.0, 'Comparing Sepal Length and Sepal Width')
```

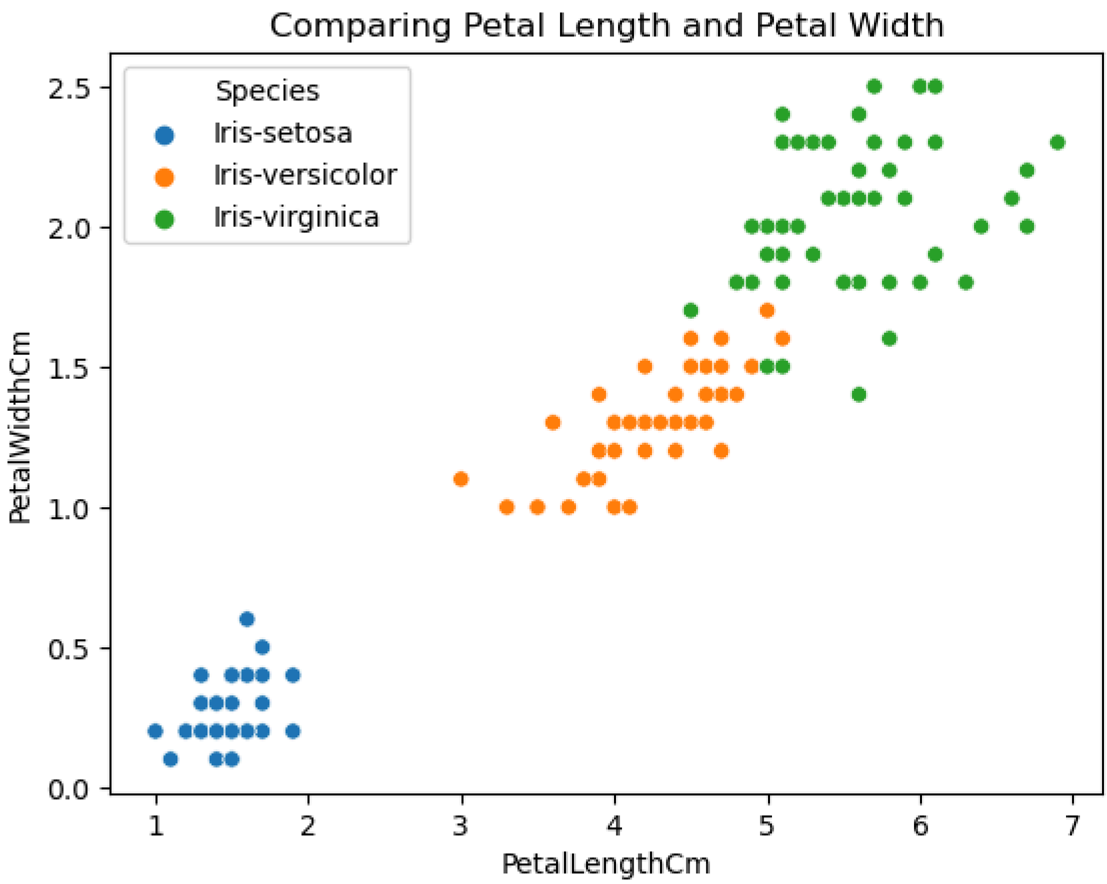


From the above plot, we can infer that –

1. Species Setosa has smaller sepal lengths but larger sepal widths.
2. Versicolor Species lies in the middle of the other two species in terms of sepal length and width
3. Species Virginica has larger sepal lengths but smaller sepal widths.

In [14]:
```
sns.scatterplot(x='PetalLengthCm',y='PetalWidthCm',hue='Species',data=data)
plt.title("Comparing Petal Length and Petal Width")
```

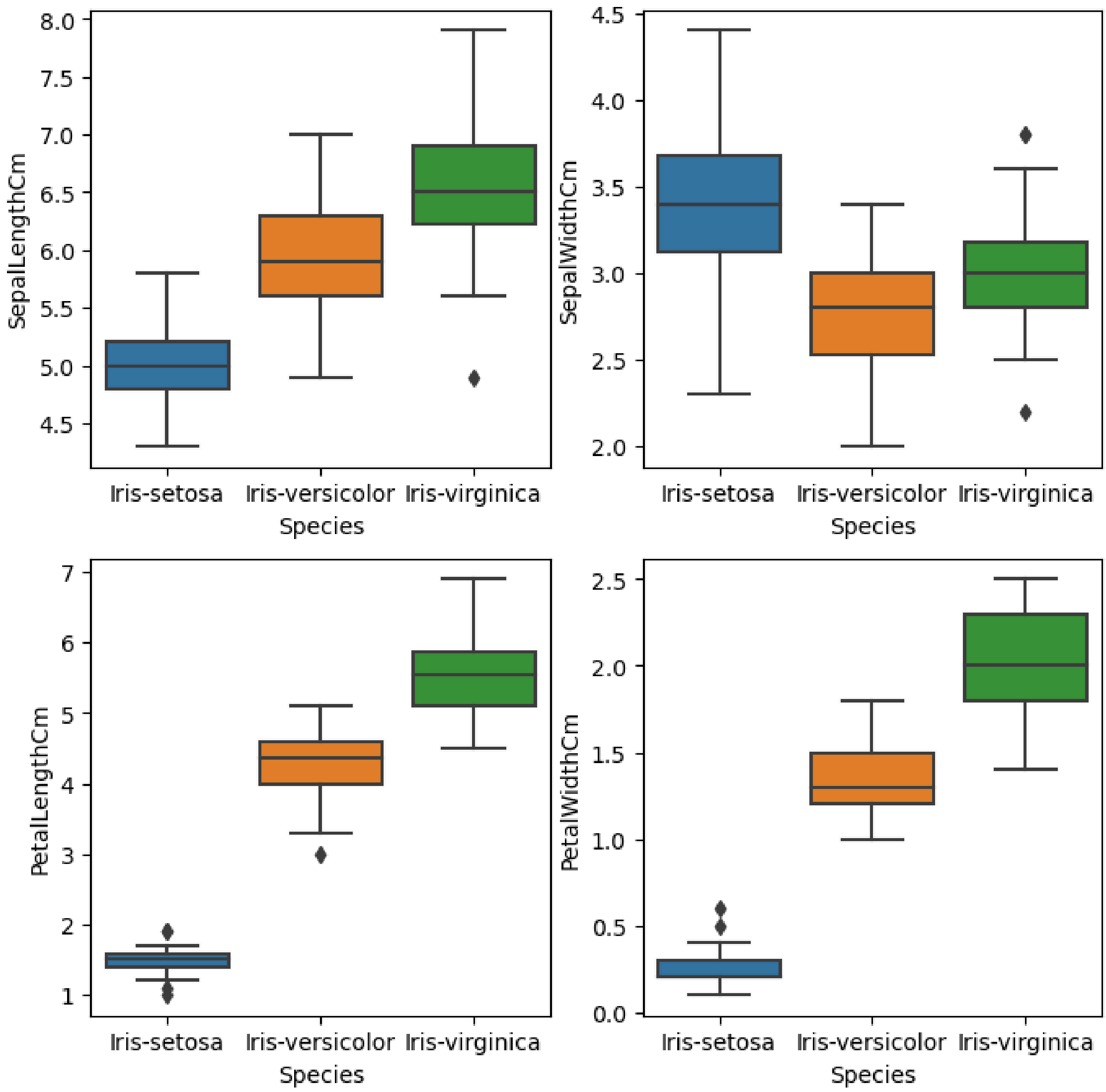
Out[14]:
```
Text(0.5, 1.0, 'Comparing Petal Length and Petal Width')
```



From the above plot, we can infer that –

1. Species Setosa has smaller petal lengths and widths.
2. Versicolor Species lies in the middle of the other two species in terms of petal length and width
3. Species Virginica has the largest of petal lengths and widths.

In [17]:
```
plt.figure(figsize=(8,8))

plt.subplot(221)
sns.boxplot(x="Species",y='SepalLengthCm', data=data)

plt.subplot(222)
sns.boxplot(x="Species", y='SepalWidthCm', data=data)

plt.subplot(223)
sns.boxplot(x="Species", y='PetalLengthCm', data=data)

plt.subplot(224)
sns.boxplot(x="Species", y='PetalWidthCm', data=data)

plt.show()
```

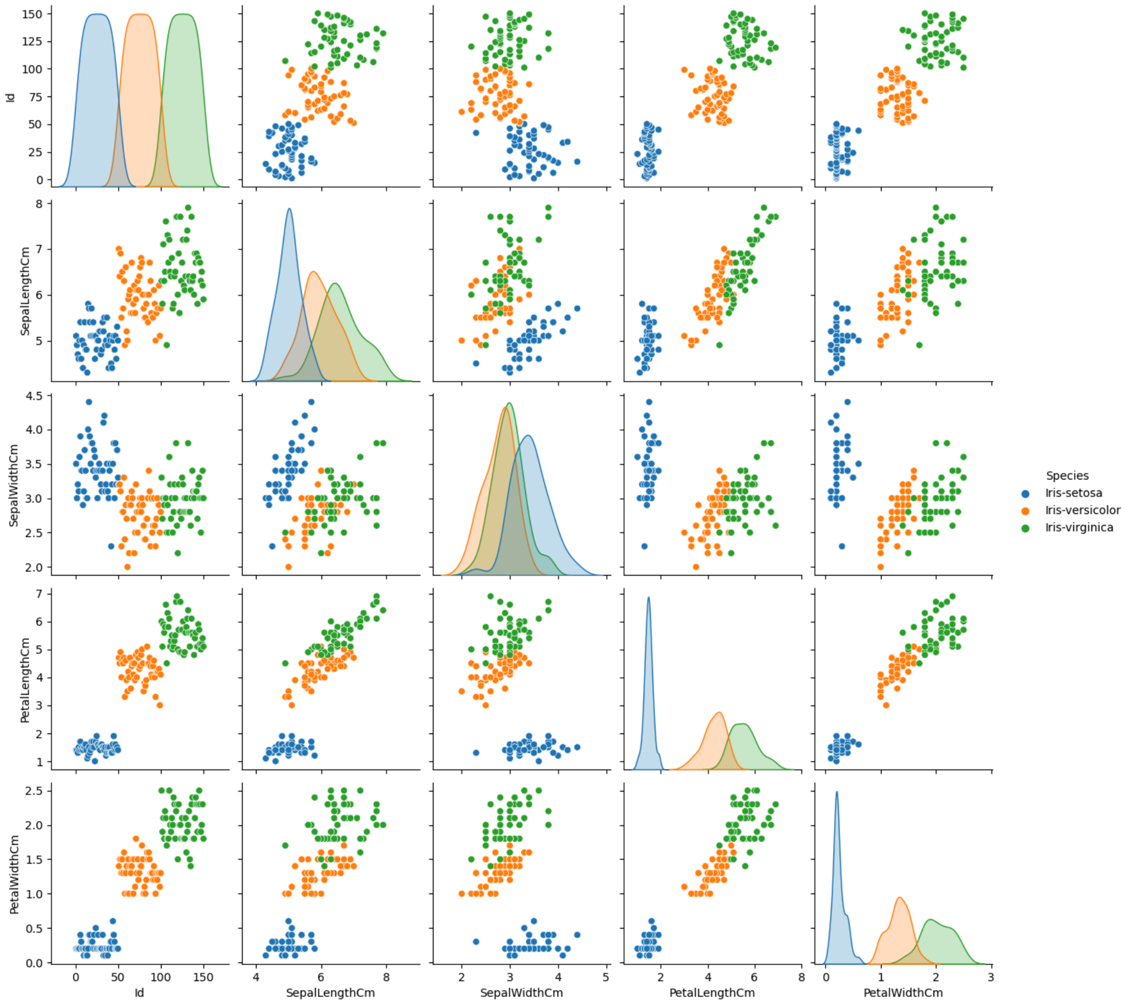


From the above graph, we can see that –

1. Species Setosa has the smallest features and less distributed with some outliers.
2. Species Versicolor has the average features.
3. Species Virginica has the highest features

In [24]:
```
sns.pairplot(data=data,hue="Species")
plt.show()
```

```
C:\Users\user\anaconda05\Lib\site-packages\seaborn\axisgrid.py:118: UserW
arning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



Data Insights:

1. High co relation between petal length and width columns.
2. Setosa has both low petal length and width
3. Versicolor has both average petal length and width
4. Virginica has both high petal length and width.
5. Sepal width for setosa is high and length is low.
6. Versicolor have average values for for sepal dimensions.
7. Virginica has small width but large sepal length

In [ ]: