Smital bhalerao

Task 1: The Iris Flower Classification ML Project

LETS GROW MORE(LGM) VIP INTERNSHIP

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

Importing Dataset

In [6]:
```python
df = pd.read_csv('C:\\Users\\Smital Bhalerao\\Desktop\\Iris.csv')
df.head()
```

Out[6]:

	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

Data Preprocessing

In [7]:
```python
# to basic info about datatypes
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]:
```python
# to display no. of samples on each class
df['Species'].value_counts()
```

Out[8]:
```
Iris-virginica     50
Iris-setosa        50
Iris-versicolor    50
Name: Species, dtype: int64
```

In [9]:
```python
# check for null values
df.isnull().sum()
```

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

In [10]:
```python
df.describe()
```

Out[10]:

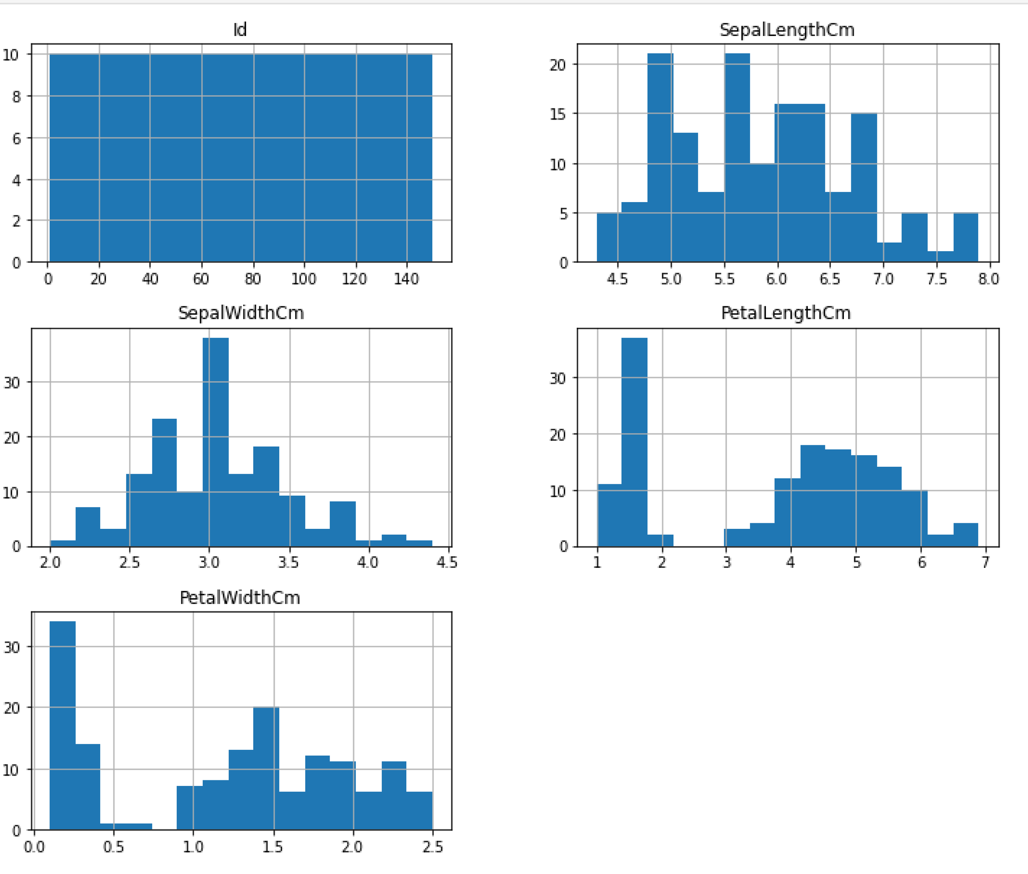
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	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 [11]:
```python
df.dtypes
```

Out[11]:
```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

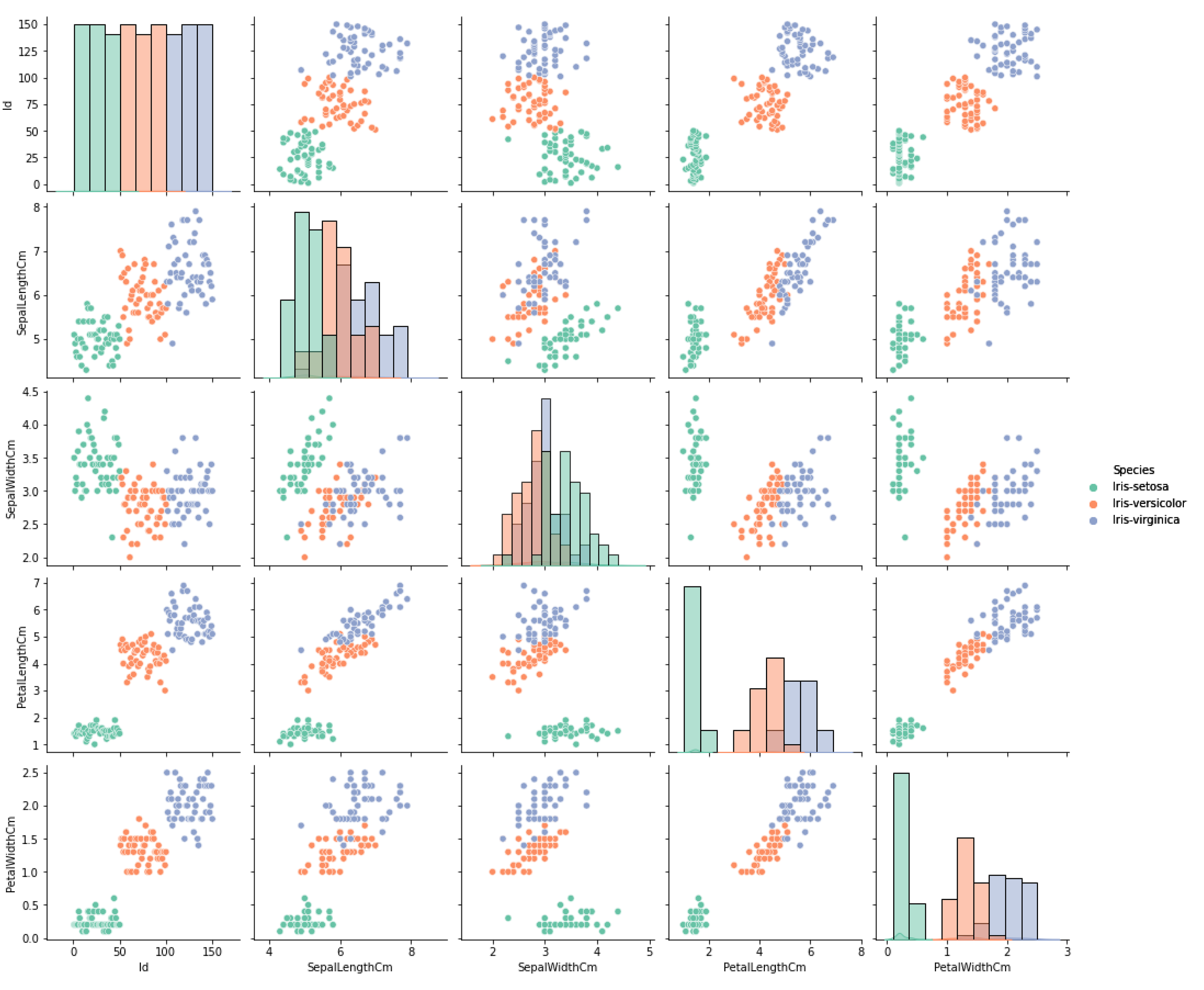
Data Visualisation

In [12]:
```python
df.hist(figsize=(12,10),bins=15)
plt.show()
```

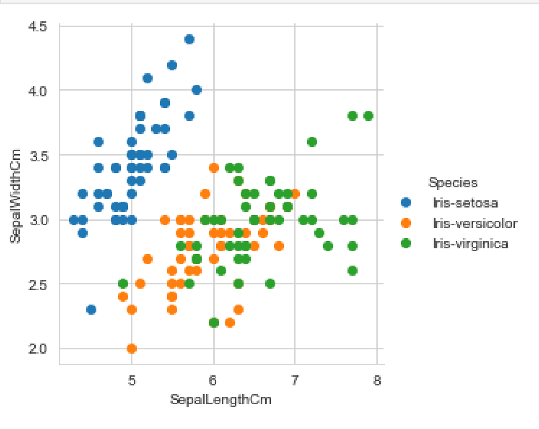


In [13]:
```python
p=sns.pairplot(df, hue='Species', palette="Set2")
p.map_diag(sns.histplot)
p.map_offdiag(sns.scatterplot)
p.add_legend()
```

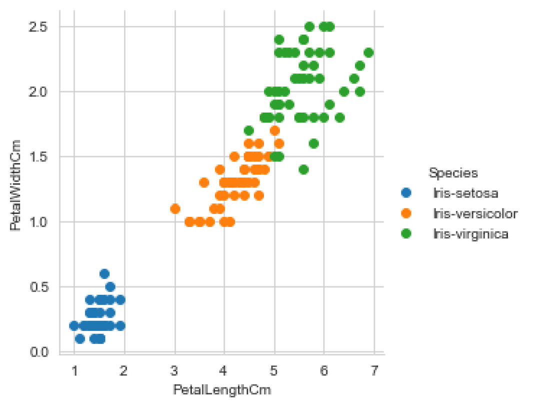
Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x187df34baf0>
```



In [15]:
```python
# scatter plot with different colour for each flower type
sns.set_style("whitegrid");
sns.FacetGrid(df, hue="Species",height=4)\
   .map(plt.scatter, "SepalLengthCm","SepalWidthCm")\
   .add_legend();
plt.show();
```

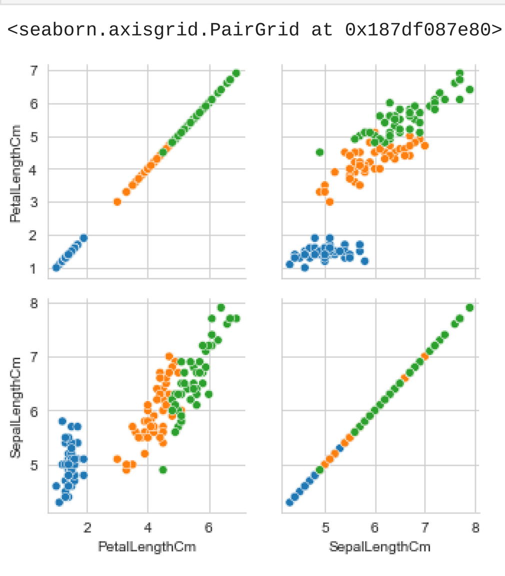


In [16]:
```python
sns.set_style("whitegrid");
sns.FacetGrid(df, hue="Species",height=4)\
   .map(plt.scatter, "PetalLengthCm","PetalWidthCm")\
   .add_legend();
plt.show();
```



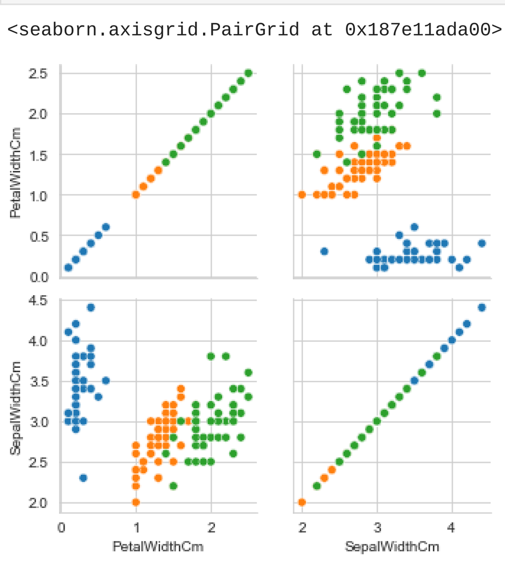
In [17]:
```python
p=sns.PairGrid(df,vars=["PetalLengthCm","SepalLengthCm"],hue="Species")
p.map(sns.scatterplot)
```

Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x187df087e80>
```



In [18]:
```python
p=sns.PairGrid(df,vars=["PetalWidthCm","SepalWidthCm"],hue="Species")
p.map(sns.scatterplot)
```

Out[18]:
```
<seaborn.axisgrid.PairGrid at 0x187e11ada00>
```



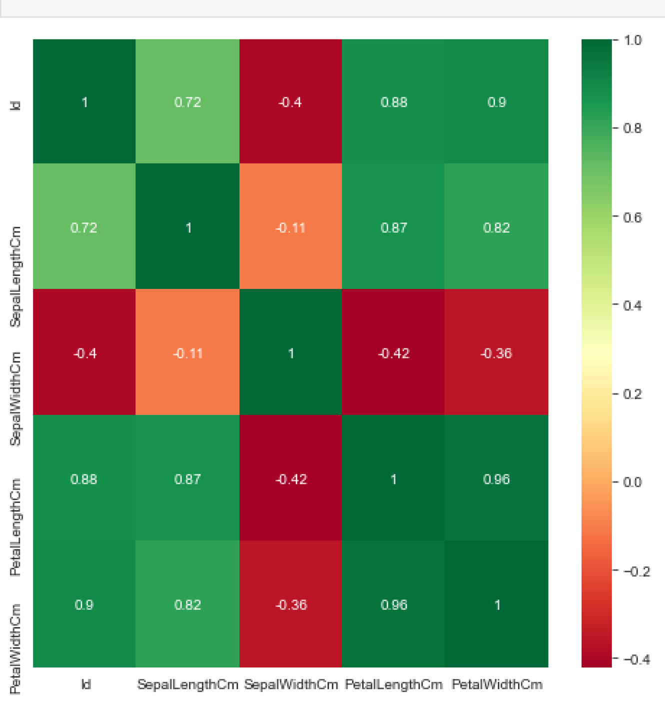
Correlation Matrix

In [19]:
```python
df.corr()
```

Out[19]:

	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 [20]:
```python
plt.figure(figsize=(8,8))
p=sns.heatmap(df.corr(),annot=True,cmap='RdYlGn')
# The sepalwidthCm feature seems to be less relevant in explaining the species as compared to others
```



In [21]:
```python
from sklearn.preprocessing import StandardScaler
features = ["SepalLengthCm","SepalWidthCm","PetalLengthCm","PetalWidthCm"]
x= df.loc[:, features].values
y=df.loc[:,['Species']].values

# standardizing the features
x= StandardScaler().fit_transform(x)
```

Splitting Data Into Training And Testing

In [22]:
```python
from sklearn.model_selection import train_test_split
y=df['Species']
x_train, x_test, y_train, y_test= train_test_split(x,y, test_size=0.40)
```

Logistic Regression

In [23]:
```python
from sklearn.linear_model import LogisticRegression
log_reg= LogisticRegression()
```

In [24]:
```python
# model training
log_reg.fit(x_train, y_train)
```

Out[24]:
```
LogisticRegression()
```

In [25]:
```python
# accuracy
print("Accuracy:")
log_reg.score(x_test, y_test)
```

```
Accuracy:
```

Out[25]:
```
0.9833333333333333
```

In [26]:
```python
log_reg.score(x,y)
```

Out[26]:
```
0.9666666666666667
```

KNN k-nearest neighbours

In [36]:
```python
from sklearn.neighbors import KNeighborsClassifier
knnmodel = KNeighborsClassifier()
```

In [37]:
```python
knnmodel.fit(x_train , y_train)
```

Out[37]:
```
KNeighborsClassifier()
```

In [38]:
```python
print("Accuracy:")
knnmodel.score(x_test , y_test)*100
```

```
Accuracy:
```

Out[38]:
```
96.66666666666667
```

In [ ]: