Oasis Infobyte

Task 1 : Iris Flower Classification

Iris flower has three species; setosa, versicolor, and virginica, which differs according to their measurements. Now assume that you have the measurements of the iris flowers according to their species, and here your task is to train a machine learning model that can learn from the measurements of the iris species and classify them.

1. Import all necessary

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

2. Import dataframe

In [2]:
```
df = pd.read_csv("iris.csv")
```

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

3. Check for Null Values

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

we don't have null values

4. Check for Dublicate row

In [5]:
```
df.duplicated().sum()
```

Out[5]:
```
0
```

we don't have dublicate row

5. Summery of data

In [6]:
```
df.describe()
```

Out[6]:

	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 [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
```

6. Check column name

In [8]:
```
df.columns
```

Out[8]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthC
m',
       'Species'],
      dtype='object')
```

7. Check the datatype

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

Out[9]:
```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

8.Shape of dataset

In [10]:
```
df.shape
```

Out[10]:
```
(150, 6)
```

9. Find Corelation of data

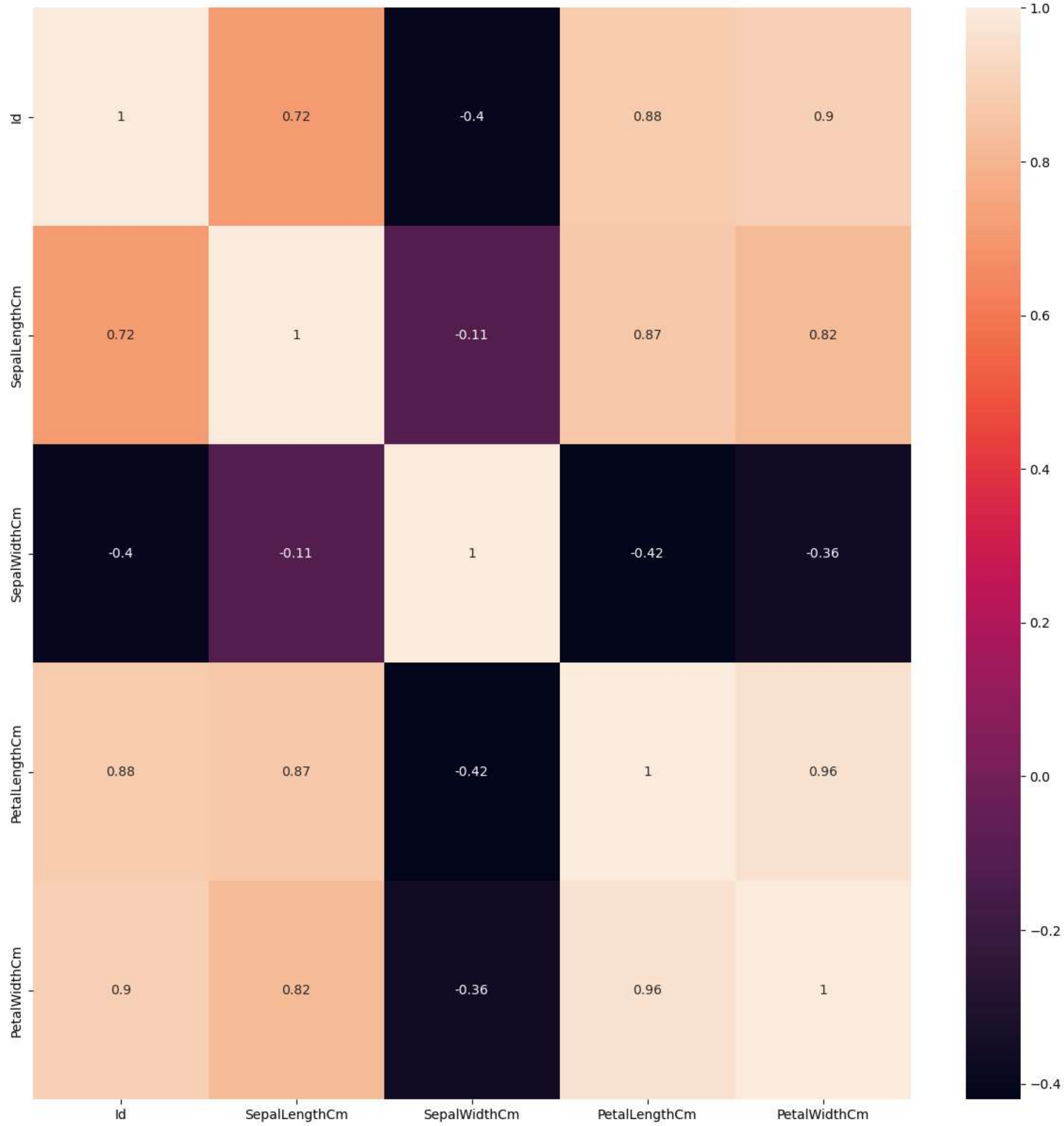
In [11]:
```
df.corr()
```

Out[11]:

	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 [12]:
```
plt.figure(figsize=(15,15))
sns.heatmap(df.corr() , annot=True)
```

Out[12]: <AxesSubplot:>



10. Analysing the 'Spices' column

In [13]:
```
df["Species"]
```

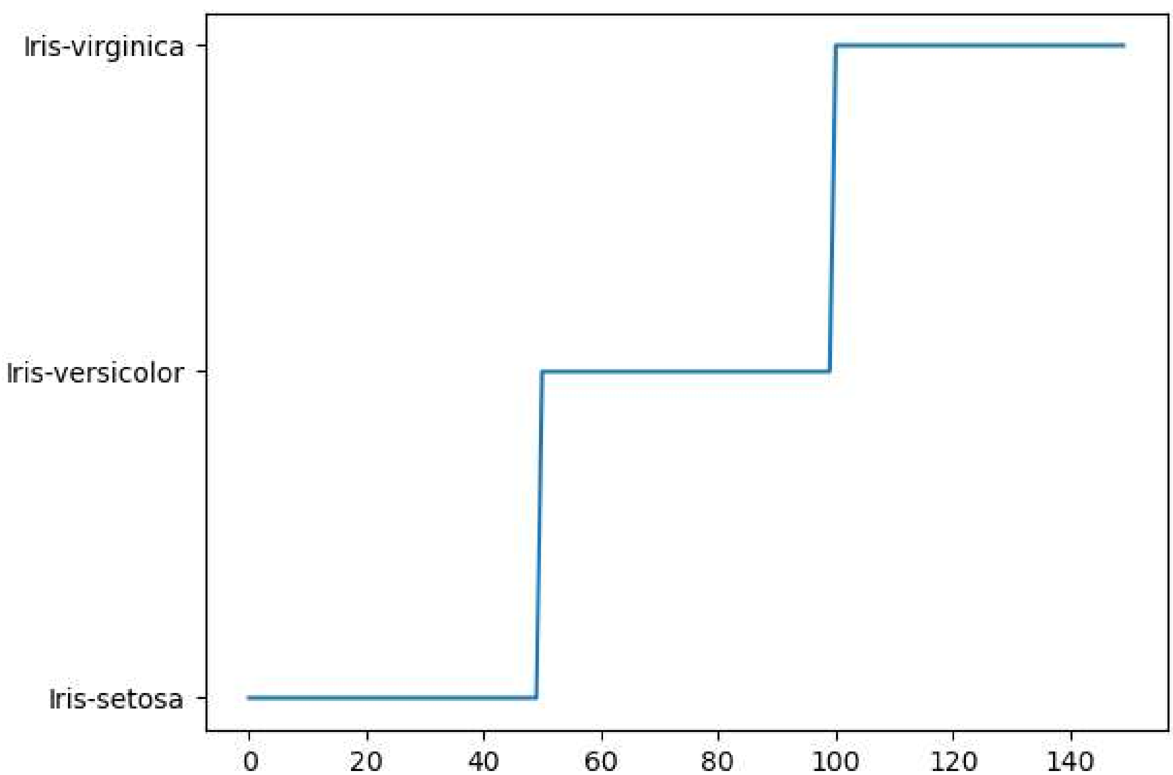
Out[13]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

In [14]:
```
df["Species"].describe()
```

Out[14]:
```
count             150
unique              3
top       Iris-setosa
freq               50
Name: Species, dtype: object
```

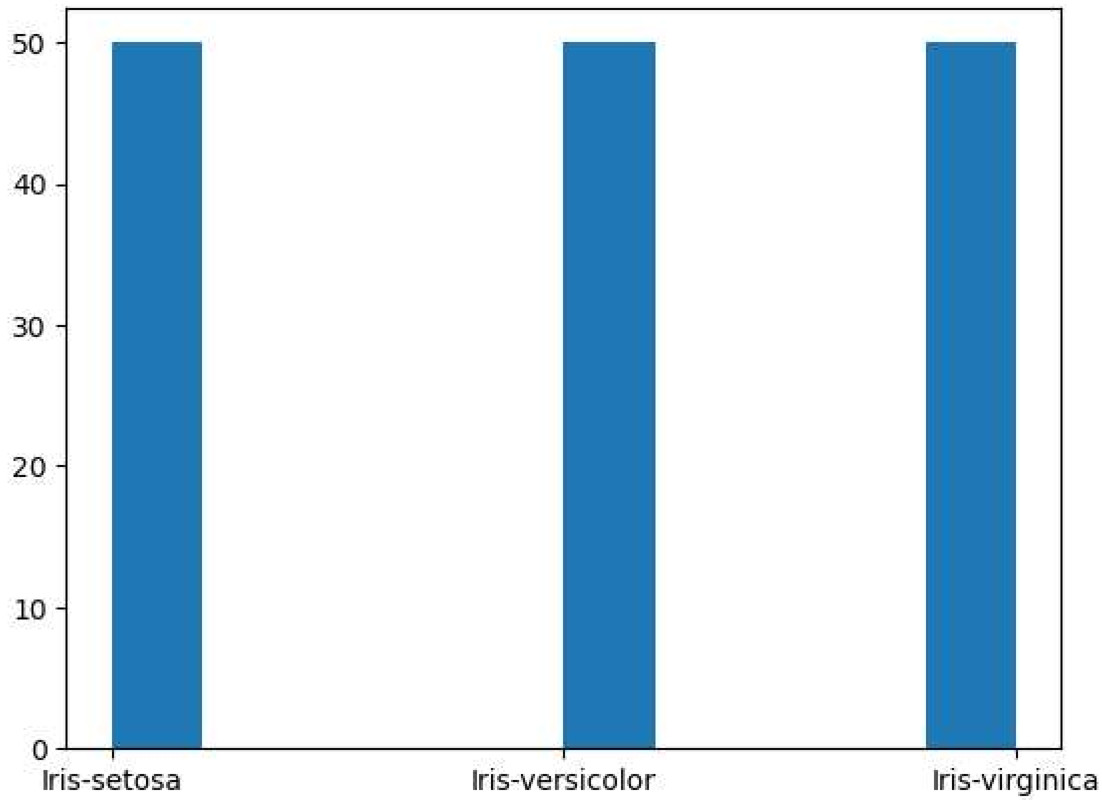
In [15]:
```
plt.plot(df["Species"])
```

Out[15]:
```
[<matplotlib.lines.Line2D at 0x1e92737f310>]
```



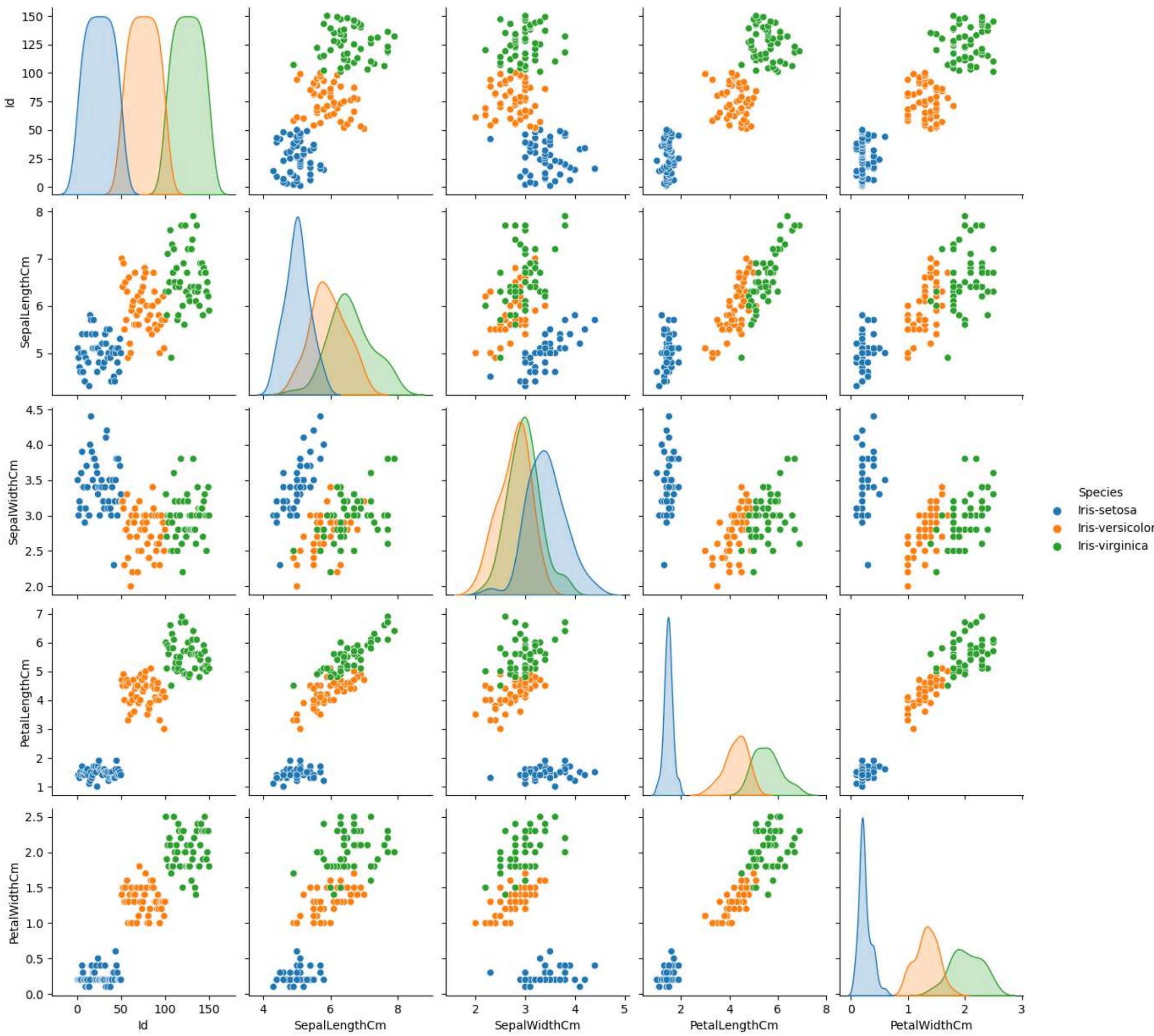
In [16]:
```
plt.hist(df["Species"])
```

Out[16]:
```
(array([50.,  0.,  0.,  0.,  0., 50.,  0.,  0.,  0., 50.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. ]),
 <BarContainer object of 10 artists>)
```



In [17]:
```
sns.pairplot(df , hue = "Species")
```

Out[17]: <seaborn.axisgrid.PairGrid at 0x1e9273cecd0>



11. Analysing the "SepalLengthCm" , "SepalWidthCm" , "PetalLengthCm" , "PetalWidthCm" column

In [18]:
```
df
```

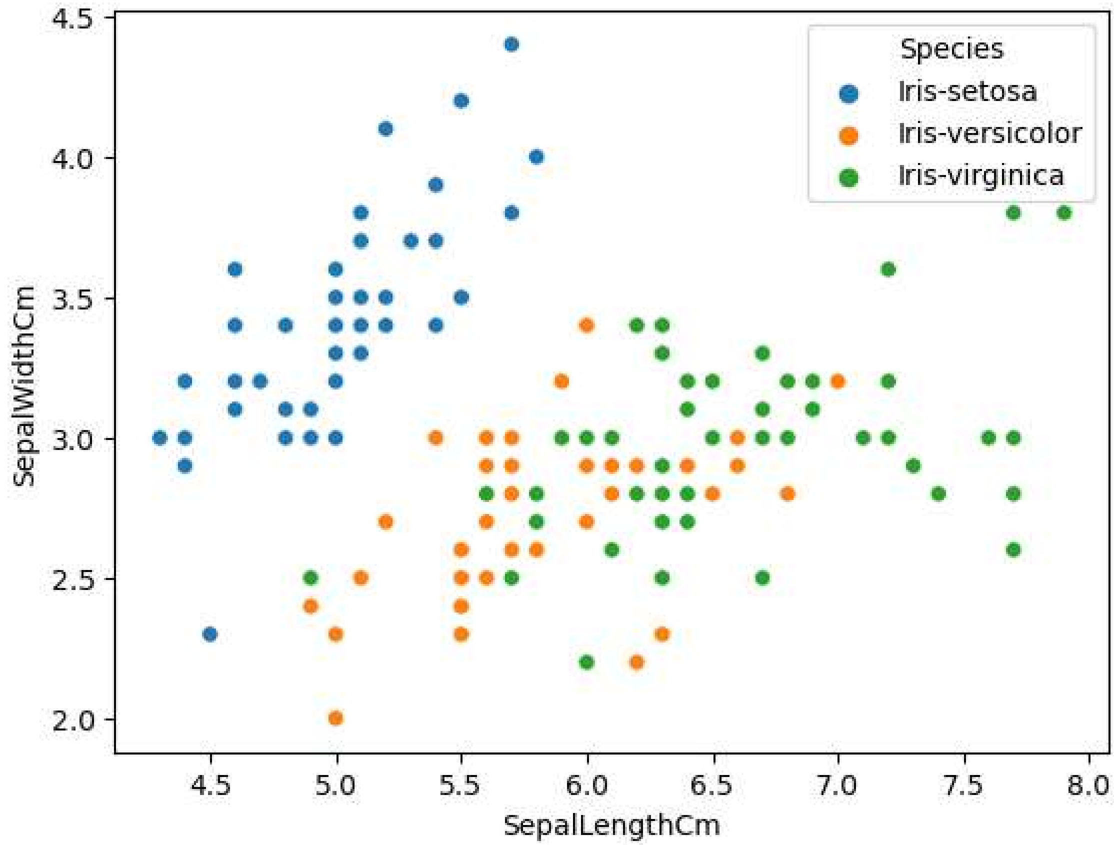
Out[18]:

	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

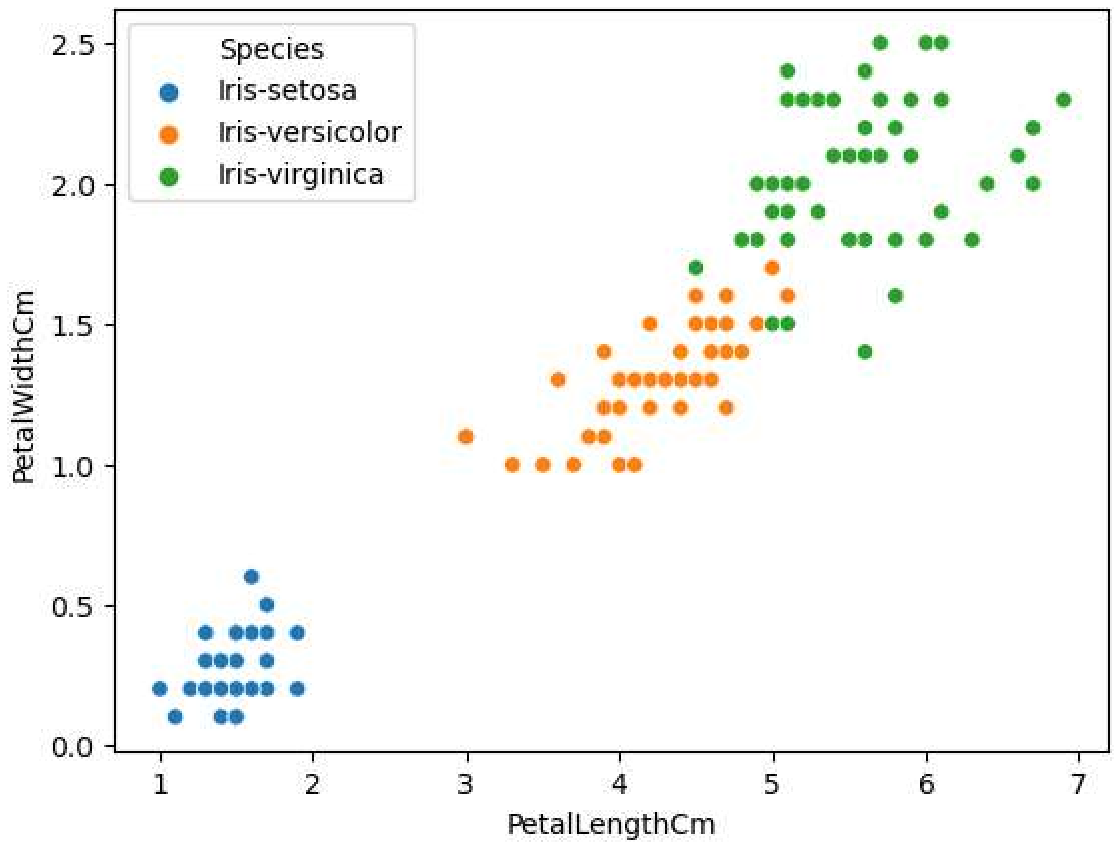
In [19]:
```
sns.scatterplot(x="SepalLengthCm" , y = "SepalWidthCm" , data = df , hue = "Spe
```

Out[19]: <AxesSubplot:xlabel='SepalLengthCm', ylabel='SepalWidthCm'>



In [20]:
```
sns.scatterplot(x="PetalLengthCm" , y = "PetalWidthCm" , data = df , hue = "Spe
```

Out[20]:
```
<AxesSubplot:xlabel='PetalLengthCm', ylabel='PetalWidthCm'>
```



12. Split the data

In [21]:
```
x = df.drop("Species" , axis = 1)
```

In [22]:
```
x
```

Out[22]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	1	5.1	3.5	1.4	0.2
1	2	4.9	3.0	1.4	0.2
2	3	4.7	3.2	1.3	0.2
3	4	4.6	3.1	1.5	0.2
4	5	5.0	3.6	1.4	0.2
...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3
146	147	6.3	2.5	5.0	1.9
147	148	6.5	3.0	5.2	2.0
148	149	6.2	3.4	5.4	2.3
149	150	5.9	3.0	5.1	1.8

150 rows × 5 columns

In [23]:
```
y = df["Species"]
```

In [24]:
```
df["Species"].unique()
```

Out[24]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [25]:
```
y.replace( { "Iris-setosa" : 1 ,"Iris-virginica" : 2  , "Iris-versicolor" : 3}
```

In [26]:
```
y
```

Out[26]:
```
0      1
1      1
2      1
3      1
4      1
      ..
145    2
146    2
147    2
148    2
149    2
Name: Species, Length: 150, dtype: int64
```

13. Let's Standardize iris data

In [27]:
```
from sklearn.model_selection import  train_test_split

x_train , x_test , y_train , y_test =  train_test_split(x,y,test_size=0.20)
```

In [28]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [29]:
```
scaler = MinMaxScaler()
```

In [30]:
```
x_scaled = scaler.fit_transform(x)
```

In [31]:
```
x_scaled
```

```
       [0.87919463, 1.        , 0.75      , 0.91525424, 0.79166667],
       [0.88590604, 0.58333333, 0.33333333, 0.77966102, 0.875     ],
       [0.89261745, 0.55555556, 0.33333333, 0.69491525, 0.58333333],
       [0.89932886, 0.5       , 0.25      , 0.77966102, 0.54166667],
       [0.90604027, 0.94444444, 0.41666667, 0.86440678, 0.91666667],
       [0.91275168, 0.55555556, 0.58333333, 0.77966102, 0.95833333],
       [0.91946309, 0.58333333, 0.45833333, 0.76271186, 0.70833333],
       [0.9261745 , 0.47222222, 0.41666667, 0.6440678 , 0.70833333],
       [0.93288591, 0.72222222, 0.45833333, 0.74576271, 0.83333333],
       [0.93959732, 0.66666667, 0.45833333, 0.77966102, 0.95833333],
       [0.94630872, 0.72222222, 0.45833333, 0.69491525, 0.91666667],
       [0.95302013, 0.41666667, 0.29166667, 0.69491525, 0.75      ],
       [0.95973154, 0.69444444, 0.5       , 0.83050847, 0.91666667],
       [0.96644295, 0.66666667, 0.54166667, 0.79661017, 1.        ],
       [0.97315436, 0.66666667, 0.41666667, 0.71186441, 0.91666667],
       [0.97986577, 0.55555556, 0.20833333, 0.6779661 , 0.75      ],
       [0.98657718, 0.61111111, 0.41666667, 0.71186441, 0.79166667],
       [0.99328859, 0.52777778, 0.58333333, 0.74576271, 0.91666667],
       [1.        , 0.44444444, 0.41666667, 0.69491525, 0.70833333]])
```

14. Apply models

In [32]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

In [33]:
```
x_train
```

Out[33]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
110	111	6.5	3.2	5.1	2.0
144	145	6.7	3.3	5.7	2.5
42	43	4.4	3.2	1.3	0.2
87	88	6.3	2.3	4.4	1.3
93	94	5.0	2.3	3.3	1.0
...	...	...	...	...	...
37	38	4.9	3.1	1.5	0.1
26	27	5.0	3.4	1.6	0.4
76	77	6.8	2.8	4.8	1.4
24	25	4.8	3.4	1.9	0.2
34	35	4.9	3.1	1.5	0.1

120 rows × 5 columns

In [34]:
```
x_test
```

Out[34]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
79	80	5.7	2.6	3.5	1.0
6	7	4.6	3.4	1.4	0.3
139	140	6.9	3.1	5.4	2.1
141	142	6.9	3.1	5.1	2.3
107	108	7.3	2.9	6.3	1.8
39	40	5.1	3.4	1.5	0.2
148	149	6.2	3.4	5.4	2.3
132	133	6.4	2.8	5.6	2.2
85	86	6.0	3.4	4.5	1.6
105	106	7.6	3.0	6.6	2.1
2	3	4.7	3.2	1.3	0.2
66	67	5.6	3.0	4.5	1.5
53	54	5.5	2.3	4.0	1.3
54	55	6.5	2.8	4.6	1.5
77	78	6.7	3.0	5.0	1.7
84	85	5.4	3.0	4.5	1.5
52	53	6.9	3.1	4.9	1.5
147	148	6.5	3.0	5.2	2.0
130	131	7.4	2.8	6.1	1.9
106	107	4.9	2.5	4.5	1.7
3	4	4.6	3.1	1.5	0.2
88	89	5.6	3.0	4.1	1.3
32	33	5.2	4.1	1.5	0.1
58	59	6.6	2.9	4.6	1.3
75	76	6.6	3.0	4.4	1.4
4	5	5.0	3.6	1.4	0.2
16	17	5.4	3.9	1.3	0.4
45	46	4.8	3.0	1.4	0.3
13	14	4.3	3.0	1.1	0.1
60	61	5.0	2.0	3.5	1.0

In [35]:
```
y_train
```

Out[35]:
```
110    2
144    2
42     1
87     3
93     3
      ..
37     1
26     1
76     3
24     1
34     1
Name: Species, Length: 120, dtype: int64
```

In [52]:
```
y_test
```

Out[52]:
```
79     3
6      1
139    2
141    2
107    2
39     1
148    2
132    2
85     3
105    2
2      1
66     3
53     3
54     3
77     3
84     3
52     3
147    2
130    2
106    2
3      1
88     3
32     1
58     3
75     3
4      1
16     1
45     1
13     1
60     3
Name: Species, dtype: int64
```

In [53]:
```
model.fit(x_train, y_train)
```

Out[53]:
```
LinearRegression()
```

In [54]:
```
y_Pred = model.predict(x_test)
```

15. check model

In [55]:
```
from sklearn.metrics import r2_score
```

In [56]:
```
r2 = r2_score(y_test,y_Pred)
```

In [68]:
```
r2
```

Out[68]: 0.5788563474209586

In [58]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

In [59]:
```
MAE = mean_absolute_error(y_test,y_Pred)
```

In [60]:
```
MAE
```

Out[60]: 0.46396734938952594

In [61]:
```
np.sqrt(MAE) # root mean squared error
```

Out[61]: 0.6811514878421143

In [62]:
```
MSE = mean_squared_error(y_test,y_Pred)
```

In [63]:
```
MSE
```

Out[63]: 0.29058912027953865

In [64]:
```
slope = model.coef_
```

In [65]:
```
slope
```

Out[65]: array([-0.01003626, -0.14817517, -0.75373742, 0.54559319, -0.1202715])

In [66]:
```
intercept =   model.intercept_
```

In [67]:
```
intercept
```

Out[67]: 4.0098134974314

In [ ]:
```

```

In [ ]: