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

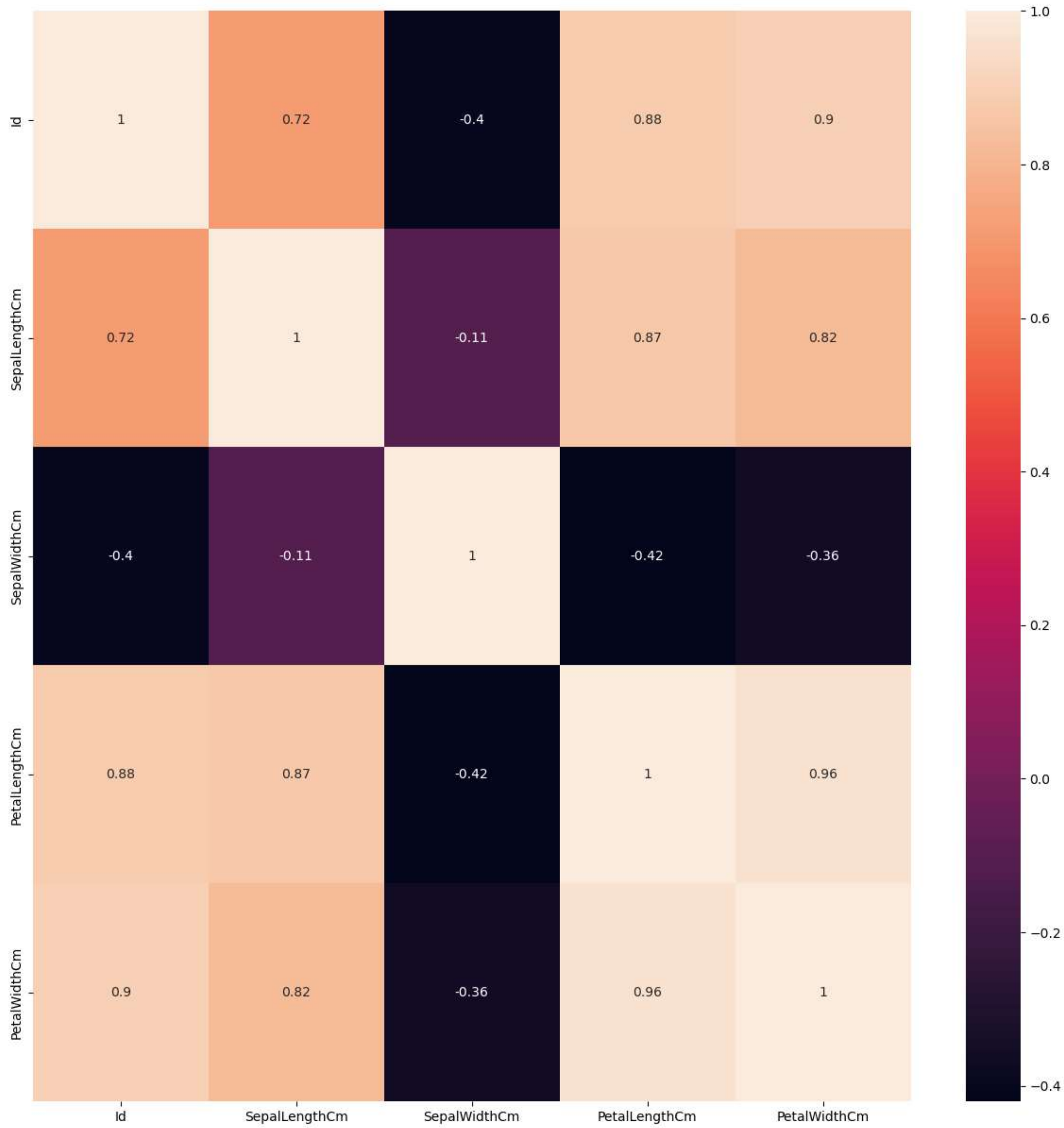
In [11]:
```
df.corr()
```

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"]
```

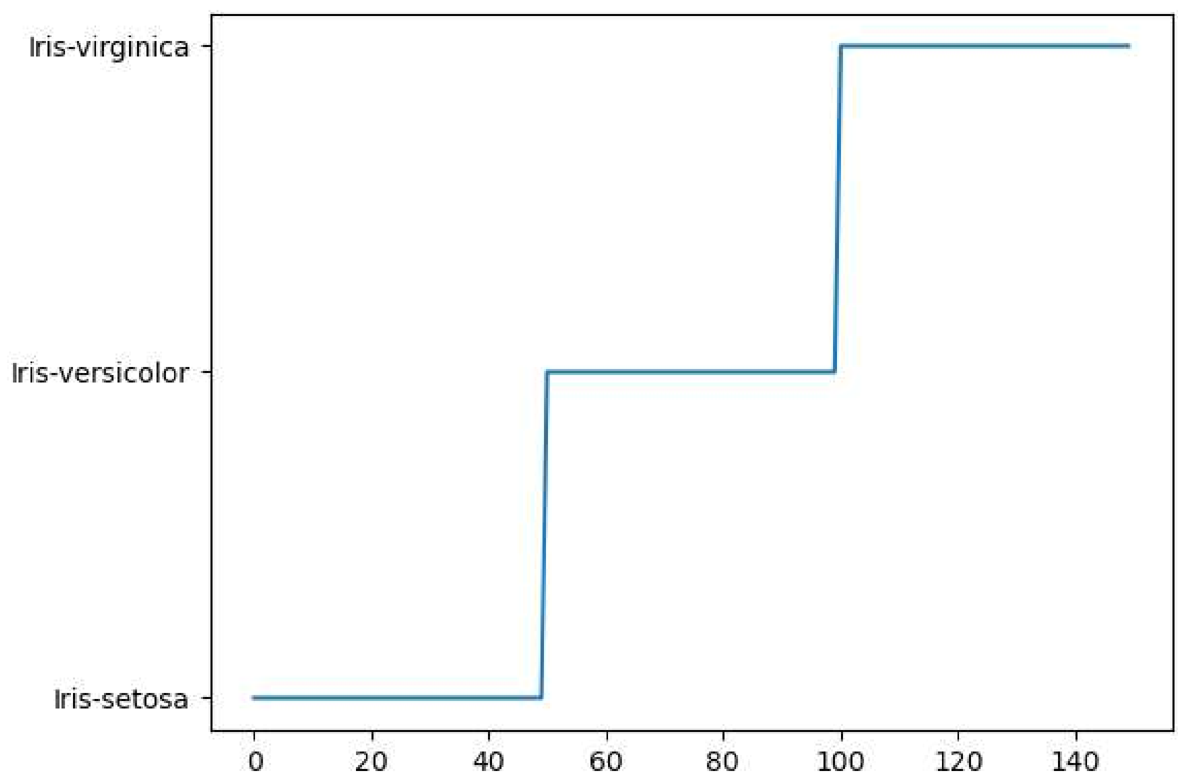
Out[13]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
             ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

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

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

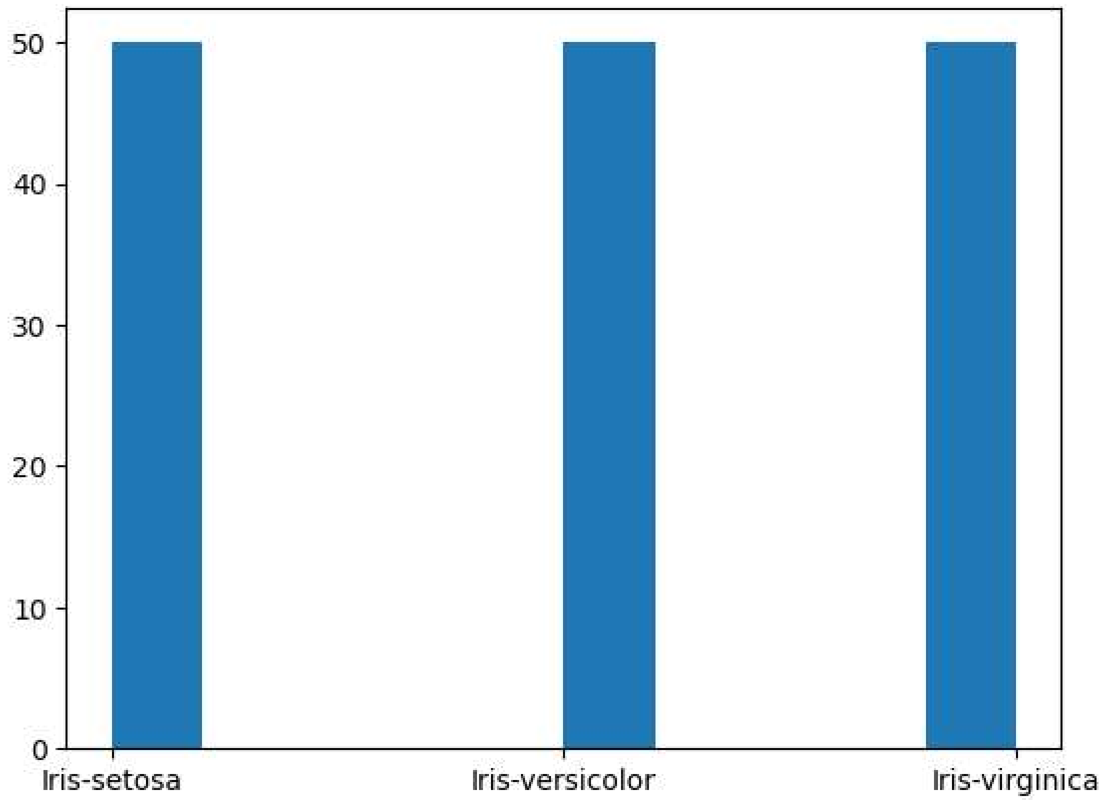
In [15]:
```
plt.plot(df["Species"])
```

Out[15]:
```
[<matplotlib.lines.Line2D at 0x21fd395c8e0>]
```

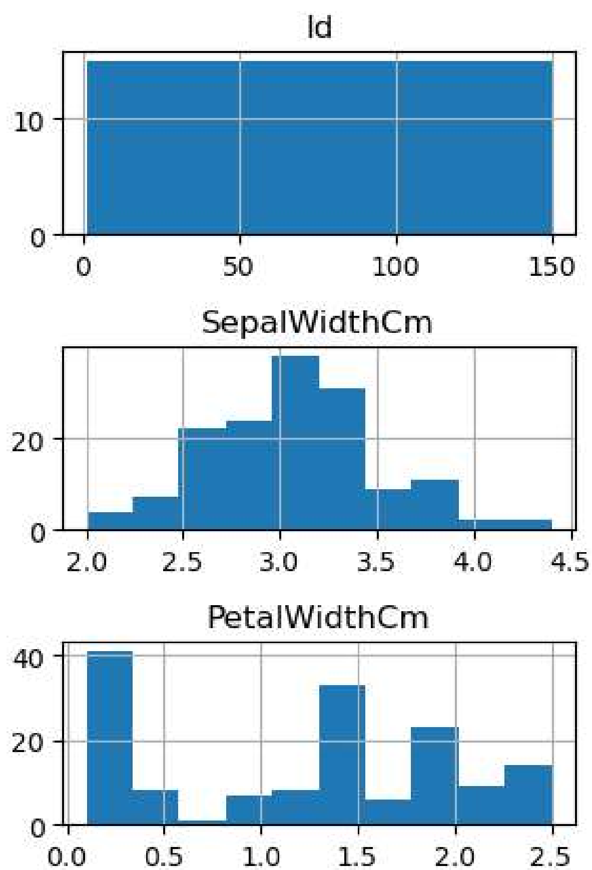


In [16]:
```
plt.hist(df["Species"])
```

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>)
```



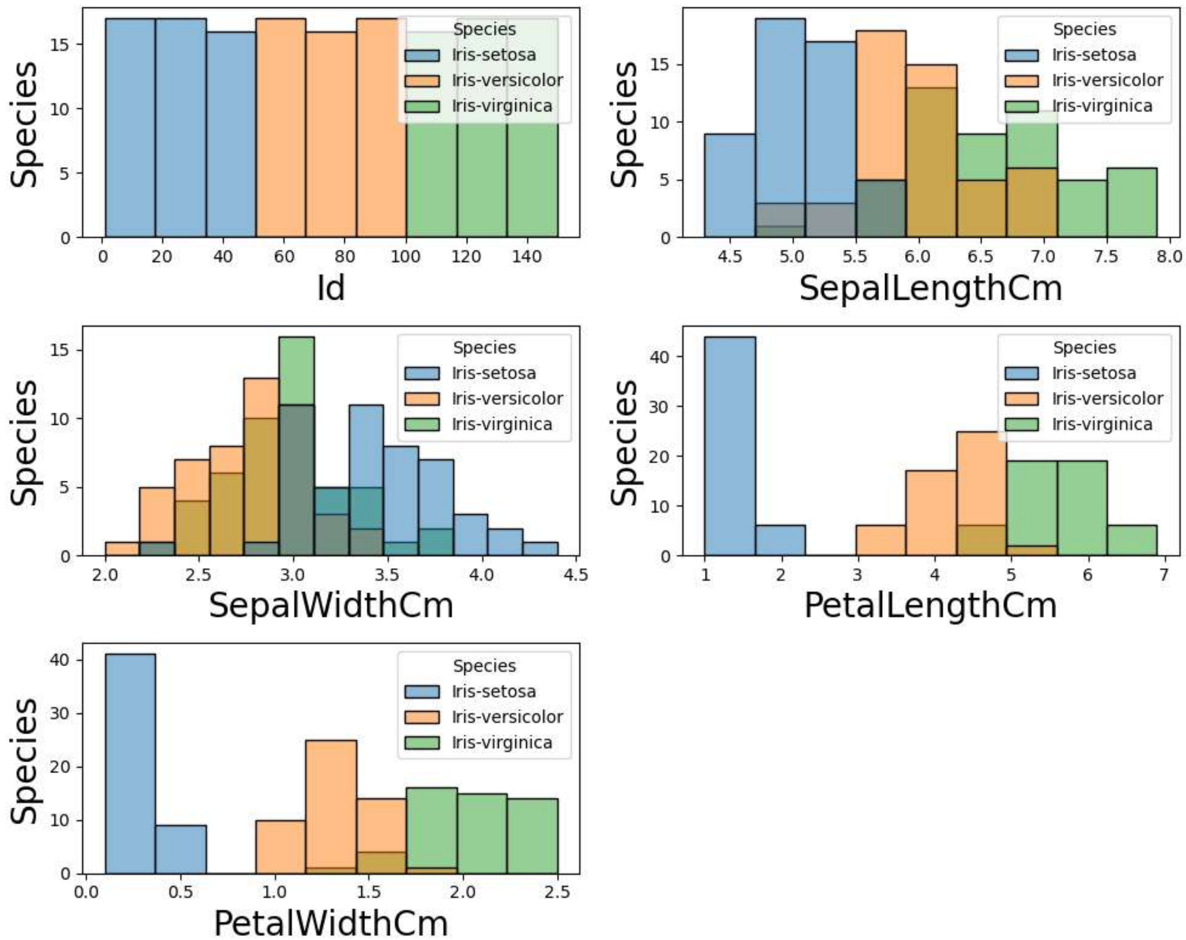
In [17]:
```
df.hist()
plt.tight_layout()
```



In [18]:
```
df.columns
```

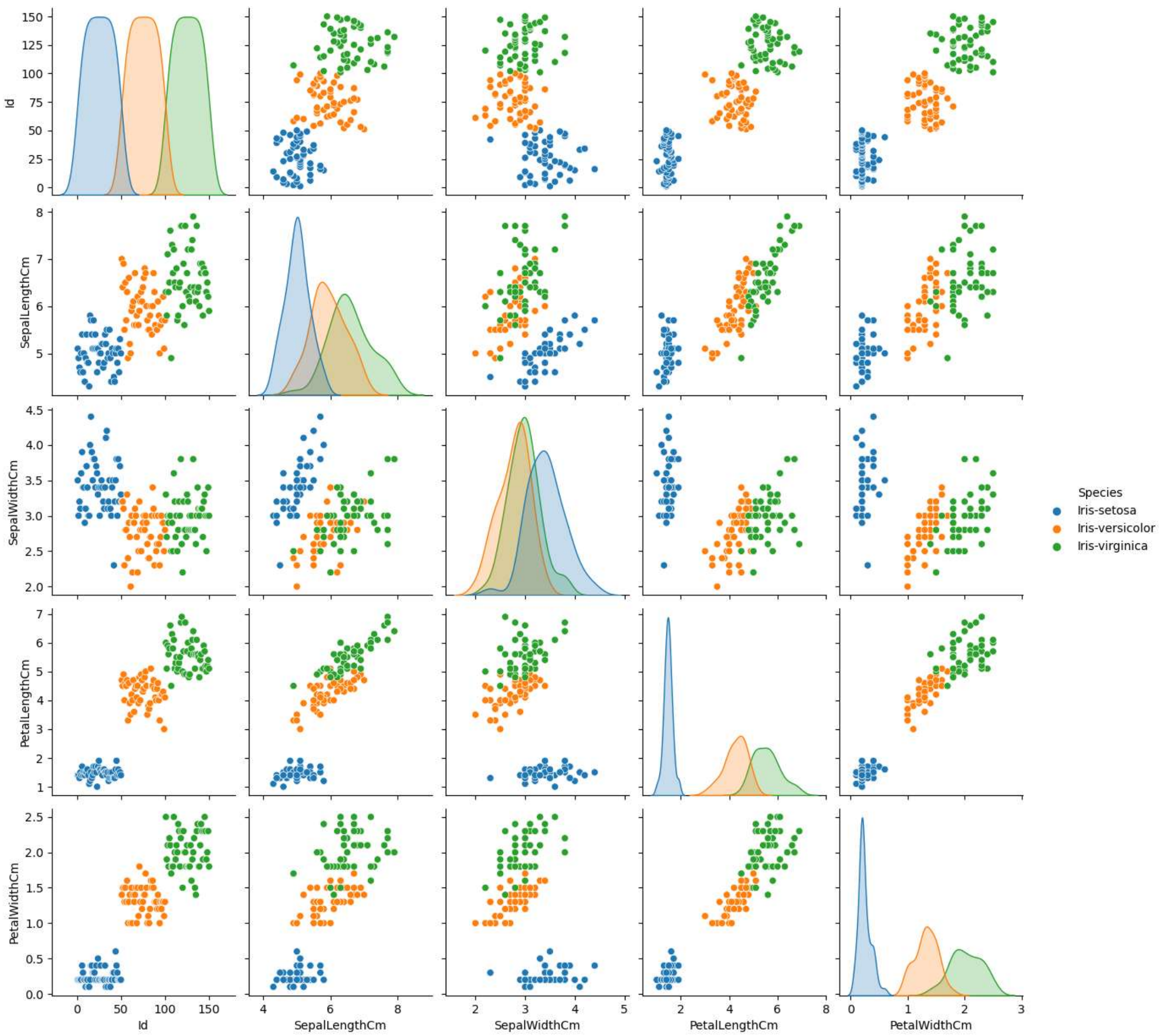
Out[18]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthC
m',
       'Species'],
      dtype='object')
```

In [19]:
```
plt.figure(figsize=(10,8))
plot  = 1
for column in df:
    if plot <=5 :
        ax = plt.subplot(3,2,plot)
        sns.histplot(x=df[column],hue=df['Species'])
        plt.xlabel(column,fontsize=20)
        plt.ylabel('Species',fontsize=20)
    plot += 1
plt.tight_layout()
```



In [20]:
```
sns.pairplot(df , hue = "Species")
```

Out[20]: <seaborn.axisgrid.PairGrid at 0x21fd409e130>



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

In [21]:
```
df
```

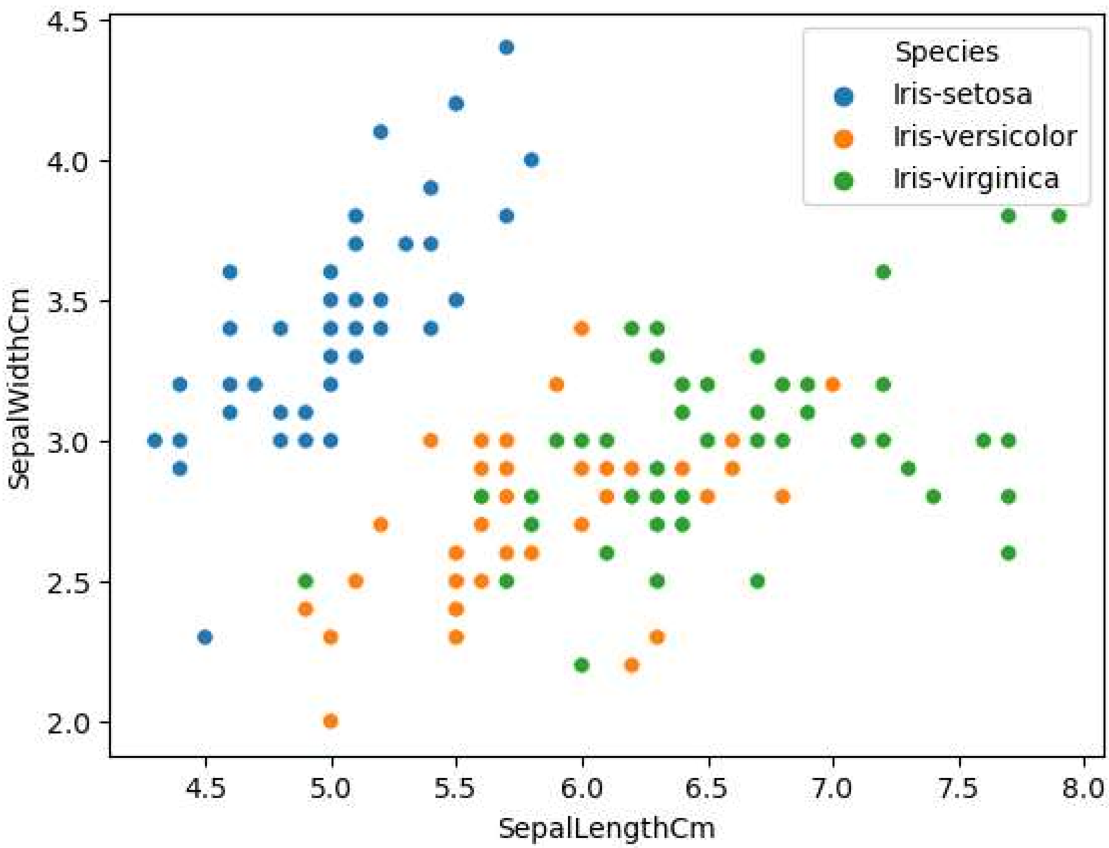
Out[21]:

	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

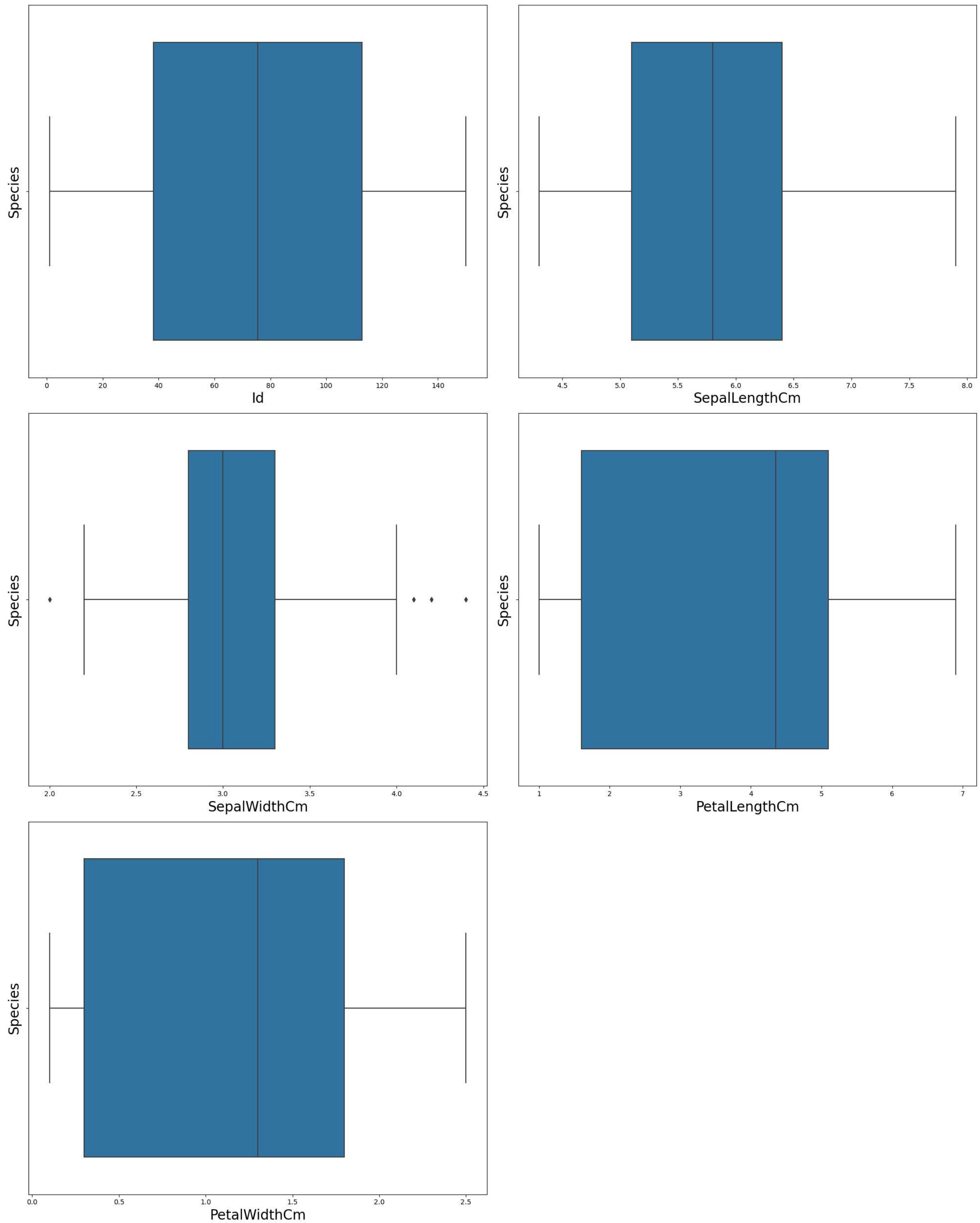
In [22]:
```
sns.scatterplot(x="SepalLengthCm" , y = "SepalWidthCm" , data = df , hue = "Spe
```

Out[22]:
```
<AxesSubplot:xlabel='SepalLengthCm', ylabel='SepalWidthCm'>
```



In [23]:
```
plt.figure(figsize=(20,25), facecolor='white')
plot  = 1

for column in df:
    if plot <= 5 :
        ax = plt.subplot(3,2,plot)
        sns.boxplot(x=df[column])
        plt.xlabel(column,fontsize=20)
        plt.ylabel('Species',fontsize=20)
    plot += 1
plt.tight_layout()
```



12. Split the data

In [24]:
```
x = df.drop("Species" , axis = 1)
```

In [25]:
```
x
```

Out[25]:

	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 [26]:
```
y = df["Species"]
```

In [27]:
```
df["Species"].unique()
```

Out[27]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [28]:
```
y.replace( { "Iris-setosa" : 1 ,"Iris-virginica" : 2  , "Iris-versicolor" : 3}
```

In [29]:
```
y
```

Out[29]:
```
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 [30]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [31]:
```
scaler = MinMaxScaler()
```

In [32]:
```
x_scaled = scaler.fit_transform(x)
```

In [33]:
```
x_scaled
```

Out[33]:
```
array([[0.        , 0.22222222, 0.625     , 0.06779661, 0.04166667],
       [0.00671141, 0.16666667, 0.41666667, 0.06779661, 0.04166667],
       [0.01342282, 0.11111111, 0.5       , 0.05084746, 0.04166667],
       [0.02013423, 0.08333333, 0.45833333, 0.08474576, 0.04166667],
       [0.02684564, 0.19444444, 0.66666667, 0.06779661, 0.04166667],
       [0.03355705, 0.30555556, 0.79166667, 0.11864407, 0.125     ],
       [0.04026846, 0.08333333, 0.58333333, 0.06779661, 0.08333333],
       [0.04697987, 0.19444444, 0.58333333, 0.08474576, 0.04166667],
       [0.05369128, 0.02777778, 0.375     , 0.06779661, 0.04166667],
       [0.06040268, 0.16666667, 0.45833333, 0.08474576, 0.        ],
       [0.06711409, 0.30555556, 0.70833333, 0.08474576, 0.04166667],
       [0.0738255 , 0.13888889, 0.58333333, 0.10169492, 0.04166667],
       [0.08053691, 0.13888889, 0.41666667, 0.06779661, 0.        ],
       [0.08724832, 0.        , 0.41666667, 0.01694915, 0.        ],
       [0.09395973, 0.41666667, 0.83333333, 0.03389831, 0.04166667],
       [0.10067114, 0.38888889, 1.        , 0.08474576, 0.125     ],
       [0.10738255, 0.30555556, 0.79166667, 0.05084746, 0.125     ],
       [0.11409396, 0.22222222, 0.625     , 0.06779661, 0.08333333],
       [0.12080537, 0.38888889, 0.75      , 0.11864407, 0.08333333],
```

In [34]:
```
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)
```

14. Apply models

In [35]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

In [36]:
```
x_train
```

Out[36]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
140	141	6.7	3.1	5.6	2.4
46	47	5.1	3.8	1.6	0.2
147	148	6.5	3.0	5.2	2.0
84	85	5.4	3.0	4.5	1.5
31	32	5.4	3.4	1.5	0.4
...	...	...	...	...	...
62	63	6.0	2.2	4.0	1.0
47	48	4.6	3.2	1.4	0.2
29	30	4.7	3.2	1.6	0.2
22	23	4.6	3.6	1.0	0.2
67	68	5.8	2.7	4.1	1.0

120 rows × 5 columns

In [37]:
```
x_test
```

Out[37]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
131	132	7.9	3.8	6.4	2.0
21	22	5.1	3.7	1.5	0.4
55	56	5.7	2.8	4.5	1.3
119	120	6.0	2.2	5.0	1.5
118	119	7.7	2.6	6.9	2.3
18	19	5.7	3.8	1.7	0.3
100	101	6.3	3.3	6.0	2.5
105	106	7.6	3.0	6.6	2.1
129	130	7.2	3.0	5.8	1.6
79	80	5.7	2.6	3.5	1.0
8	9	4.4	2.9	1.4	0.2
126	127	6.2	2.8	4.8	1.8
88	89	5.6	3.0	4.1	1.3
115	116	6.4	3.2	5.3	2.3
72	73	6.3	2.5	4.9	1.5
89	90	5.5	2.5	4.0	1.3
149	150	5.9	3.0	5.1	1.8
114	115	5.8	2.8	5.1	2.4
49	50	5.0	3.3	1.4	0.2
76	77	6.8	2.8	4.8	1.4
91	92	6.1	3.0	4.6	1.4
34	35	4.9	3.1	1.5	0.1
139	140	6.9	3.1	5.4	2.1
70	71	5.9	3.2	4.8	1.8
117	118	7.7	3.8	6.7	2.2
52	53	6.9	3.1	4.9	1.5
56	57	6.3	3.3	4.7	1.6
68	69	6.2	2.2	4.5	1.5
94	95	5.6	2.7	4.2	1.3
5	6	5.4	3.9	1.7	0.4

In [38]:
```
y_train
```

Out[38]:
```
140    2
46     1
147    2
84     3
31     1
      ..
62     3
47     1
29     1
22     1
67     3
Name: Species, Length: 120, dtype: int64
```

In [39]:
```
y_test
```

Out[39]:
```
131    2
21     1
55     3
119    2
118    2
18     1
100    2
105    2
129    2
79     3
8      1
126    2
88     3
115    2
72     3
89     3
149    2
114    2
49     1
76     3
91     3
34     1
139    2
70     3
117    2
52     3
56     3
68     3
94     3
5      1
Name: Species, dtype: int64
```

In [40]:
```
model.fit(x_train, y_train)
```

Out[40]:
```
LinearRegression()
```

In [41]:
```
y_Pred = model.predict(x_test)
```

15. check model

In [42]:
```
from sklearn.metrics import r2_score
```

In [43]:
```
r2 = r2_score(y_test,y_Pred)
```

In [44]:
```
r2
```

Out[44]:
```
0.48258673299846677
```

In [45]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

In [46]:
```
MAE = mean_absolute_error(y_test,y_Pred)
```

In [47]:
```
MAE
```

Out[47]:
```
0.4150259873184949
```

In [48]:
```
np.sqrt(MAE) # root mean squared error
```

Out[48]:
```
0.6442251060914149
```

In [49]:
```
MSE = mean_squared_error(y_test,y_Pred)
```

In [50]:
```
MSE
```

Out[50]:
```
0.2897514295208587
```

In [51]:
```
slope = model.coef_
```

In [52]:
```
slope
```

Out[52]:
```
array([-0.01045751, -0.07598971, -0.74512909,  0.48971189,  0.0146112 ])
```

In [53]:
```
intercept =   model.intercept_
```

In [54]:
```
intercept
```

Out[54]:
```
3.6555284800386785
```

16. result summary

In [55]:
```
import statsmodels.api as sm
x_train_Sm =sm.add_constant(x_train)
x_train_Sm =sm.add_constant(x_train)

ls=sm.OLS(y_train,x_train).fit()
print(ls.summary())
```

```
                            OLS Regression Results
==============================================================================
==========
Dep. Variable:                Species   R-squared (uncentered):
0.926
Model:                            OLS   Adj. R-squared (uncentered):
0.923
Method:                 Least Squares   F-statistic:
288.6
Date:                Thu, 06 Apr 2023   Prob (F-statistic):
2.51e-63
Time:                        15:50:42   Log-Likelihood:
-103.92
No. Observations:                 120   AIC:
217.8
Df Residuals:                     115   BIC:
231.8
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
====
                    coef    std err          t      P>|t|      [0.025      0.
975]
------------------------------------------------------------------------------
----
Id               -0.0066      0.003     -2.143      0.034      -0.013      -
0.000
SepalLengthCm     0.4207      0.151      2.790      0.006       0.122
0.719
SepalWidthCm     -0.3978      0.176     -2.255      0.026      -0.747      -
0.048
PetalLengthCm     0.4925      0.172      2.863      0.005       0.152
0.833
PetalWidthCm     -0.5042      0.305     -1.652      0.101      -1.109
0.100
==============================================================================
=
Omnibus:                        1.109   Durbin-Watson:                   1.52
6
Prob(Omnibus):                  0.574   Jarque-Bera (JB):                0.72
7
Skew:                           0.168   Prob(JB):                        0.69
5
Kurtosis:                       3.179   Cond. No.                         56
9.
==============================================================================
=

Notes:
[1] R² is computed without centering (uncentered) since the model does not co
ntain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
```

In [ ]:

In [ ]: