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

In [3]:
```
iris = pd.read_csv(r"C:\Users\user\Downloads\iris.data",header=None)
iris
```

Out[3]:

	0	1	2	3	4
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [5]:
```
iris.columns = ['SL','SW','PL','PW','Flower']
iris.head()
```

Out[5]:

	SL	SW	PL	PW	Flower
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 [7]:
```
iris.isnull().sum()
```

Out[7]:
```
SL        0
SW        0
PL        0
PW        0
Flower    0
dtype: int64
```

In [11]:
```
iris.dtypes
```

Out[11]:
```
SL         float64
SW         float64
PL         float64
PW         float64
Flower      object
dtype: object
```

In [13]:
```
for i in iris.columns:
    print(i,':','\n',iris[i].unique(),'\n')
```

```
SL :
 [5.1 4.9 4.7 4.6 5.  5.4 4.4 4.8 4.3 5.8 5.7 5.2 5.5 4.5 5.3 7.  6.4 6.9
 6.5 6.3 6.6 5.9 6.  6.1 5.6 6.7 6.2 6.8 7.1 7.6 7.3 7.2 7.7 7.4 7.9]

SW :
 [3.5 3.  3.2 3.1 3.6 3.9 3.4 2.9 3.7 4.  4.4 3.8 3.3 4.1 4.2 2.3 2.8 2.4
 2.7 2.  2.2 2.5 2.6]

PL :
 [1.4 1.3 1.5 1.7 1.6 1.1 1.2 1.  1.9 4.7 4.5 4.9 4.  4.6 3.3 3.9 3.5 4.2
 3.6 4.4 4.1 4.8 4.3 5.  3.8 3.7 5.1 3.  6.  5.9 5.6 5.8 6.6 6.3 6.1 5.3
 5.5 6.7 6.9 5.7 6.4 5.4 5.2]

PW :
 [0.2 0.4 0.3 0.1 0.5 0.6 1.4 1.5 1.3 1.6 1.  1.1 1.8 1.2 1.7 2.5 1.9 2.1
 2.2 2.  2.4 2.3]

Flower :
 ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

In [17]:
```
iris.shape
```

Out[17]:
```
(150, 5)
```

In [15]:
```
iris.describe()
```

Out[15]:

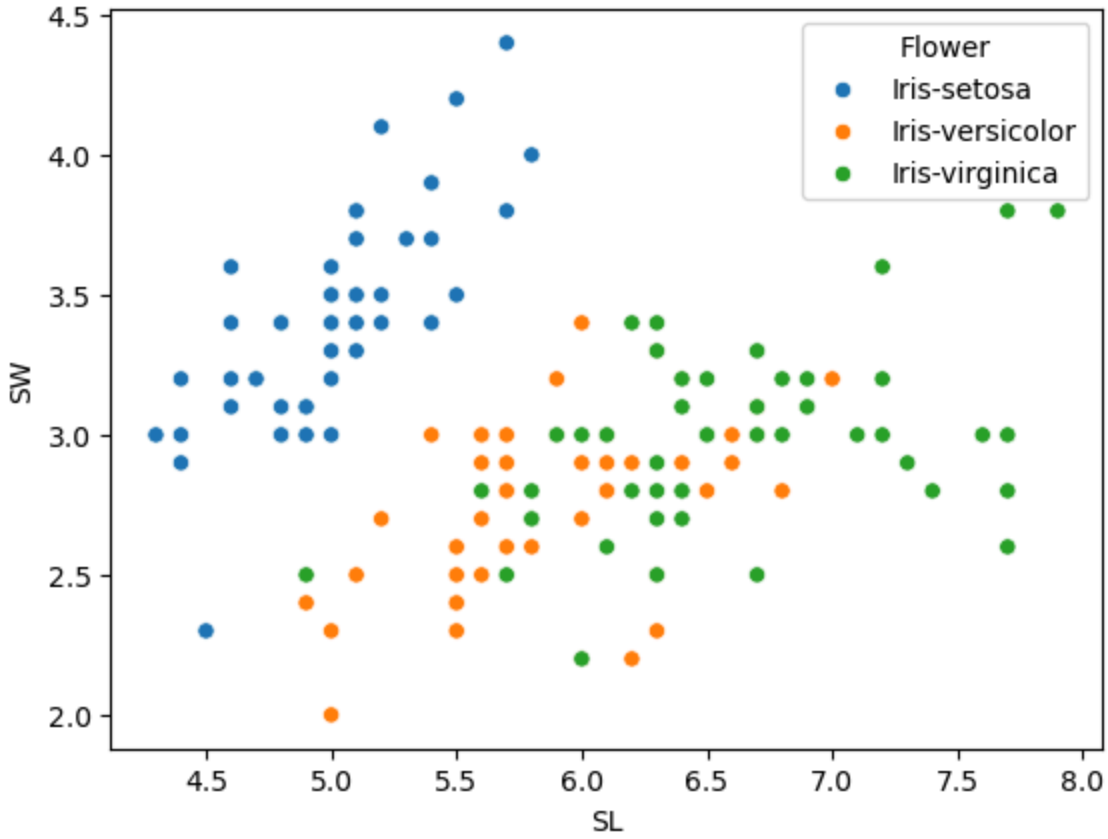
	SL	SW	PL	PW
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [21]:
```
iris.groupby('Flower').mean()
```

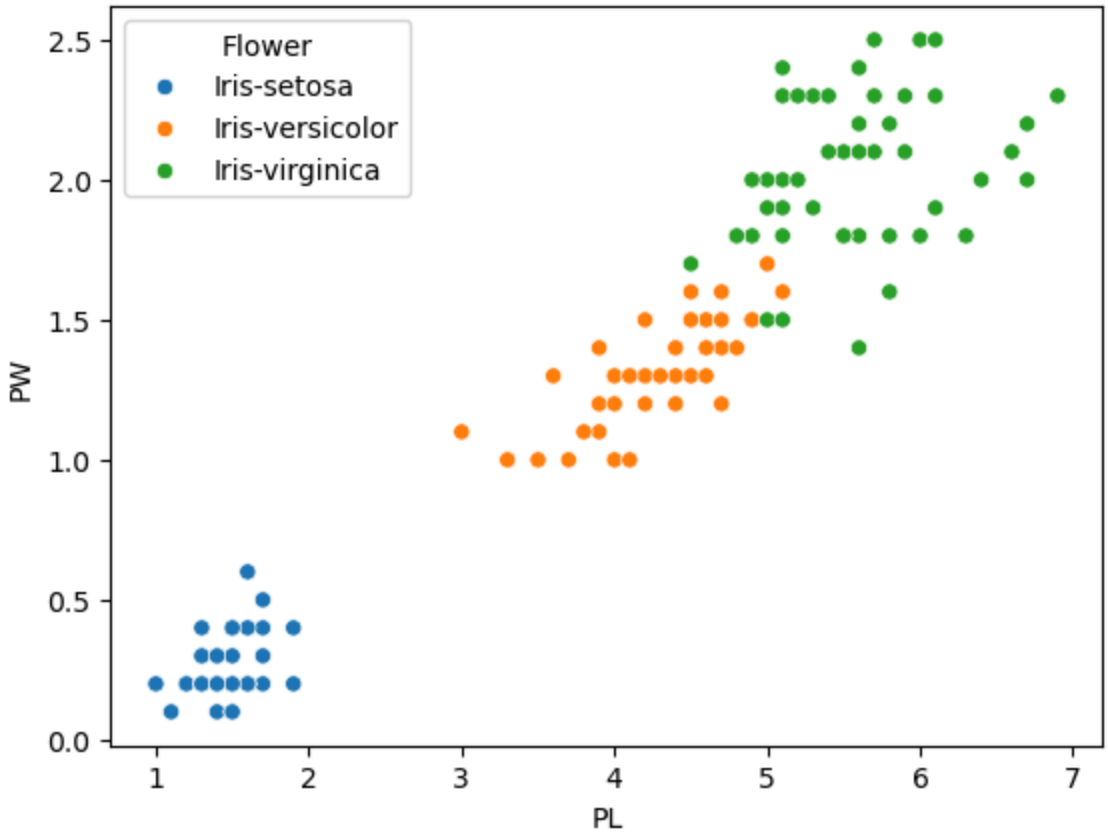
Out[21]:

Flower	SL	SW	PL	PW
Iris-setosa	5.006	3.418	1.464	0.244
Iris-versicolor	5.936	2.770	4.260	1.326
Iris-virginica	6.588	2.974	5.552	2.026

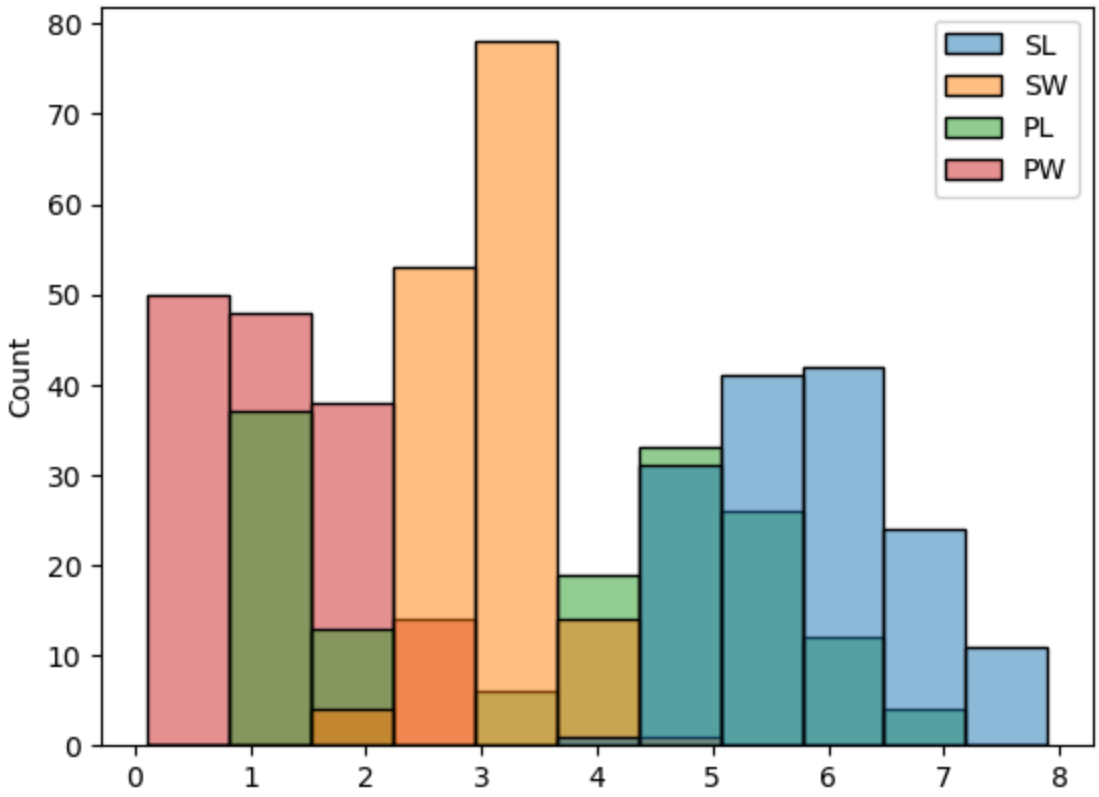
In [25]:
```
sns.scatterplot(x='SL', y='SW', hue='Flower', data=iris)
plt.show()
```



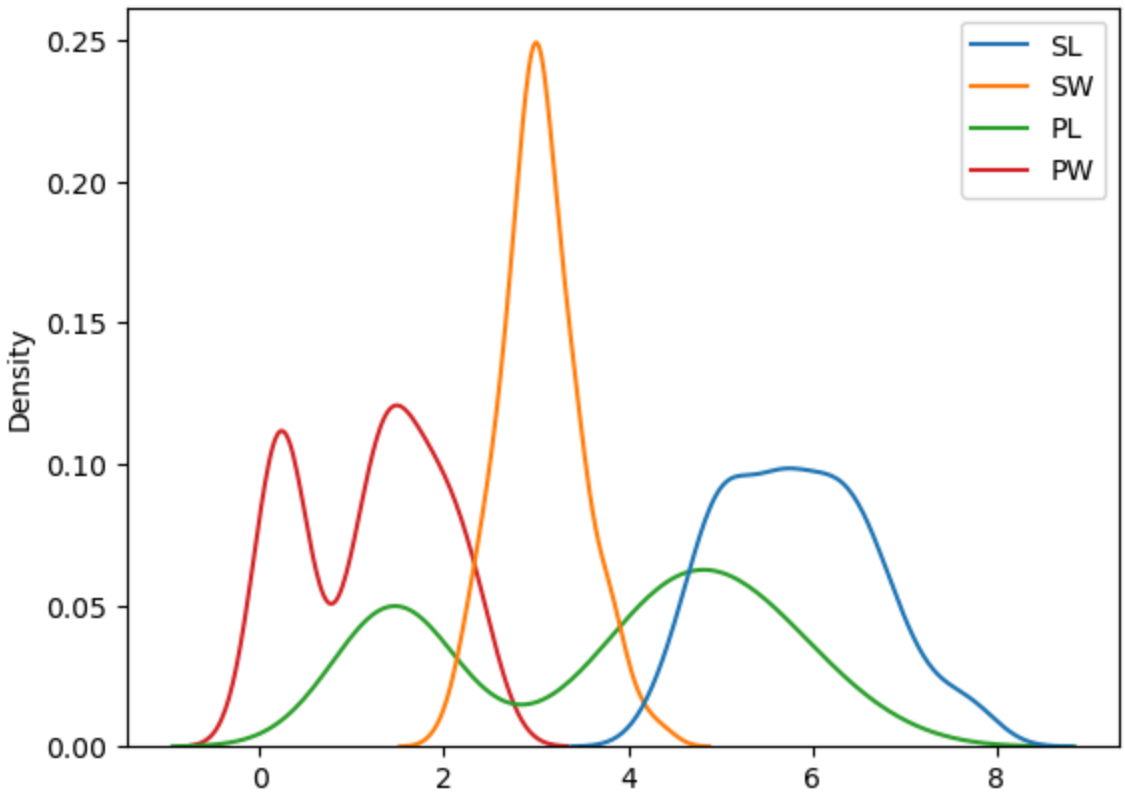
In [27]:
```
sns.scatterplot(x='PL', y='PW', hue='Flower', data=iris)
plt.show()
```



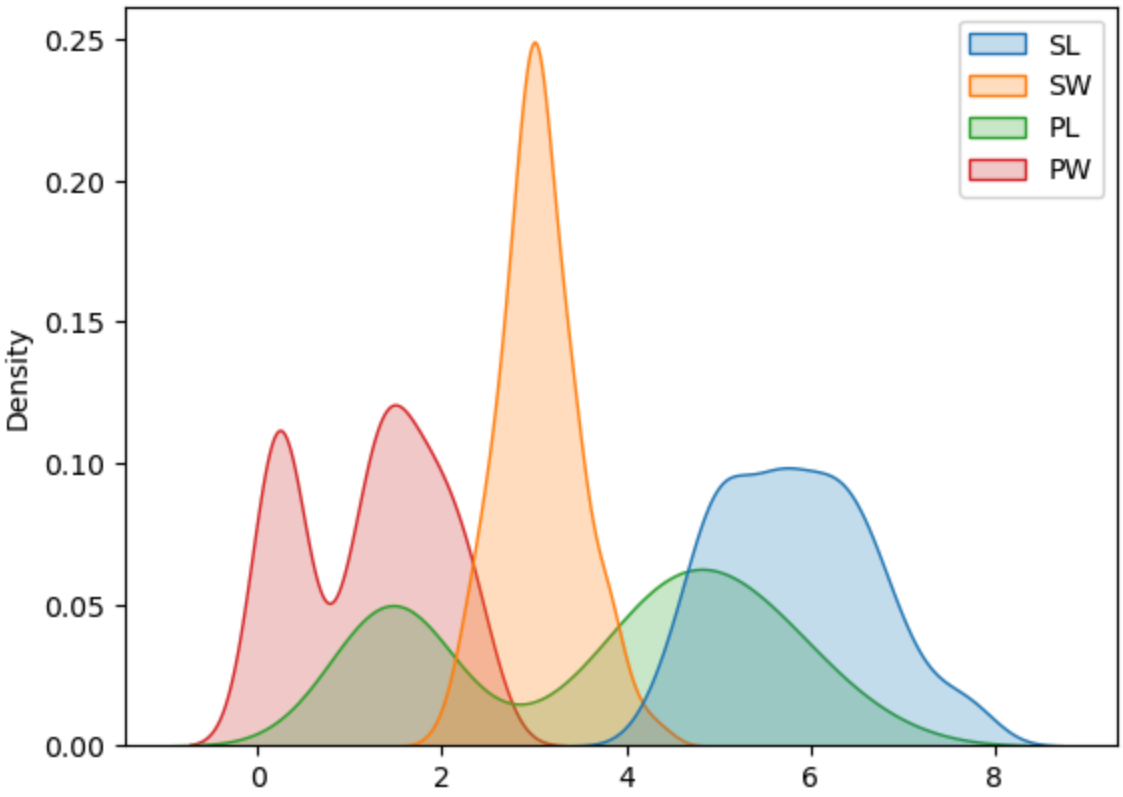
In [29]:
```
sns.histplot(data=iris)
plt.show()
```



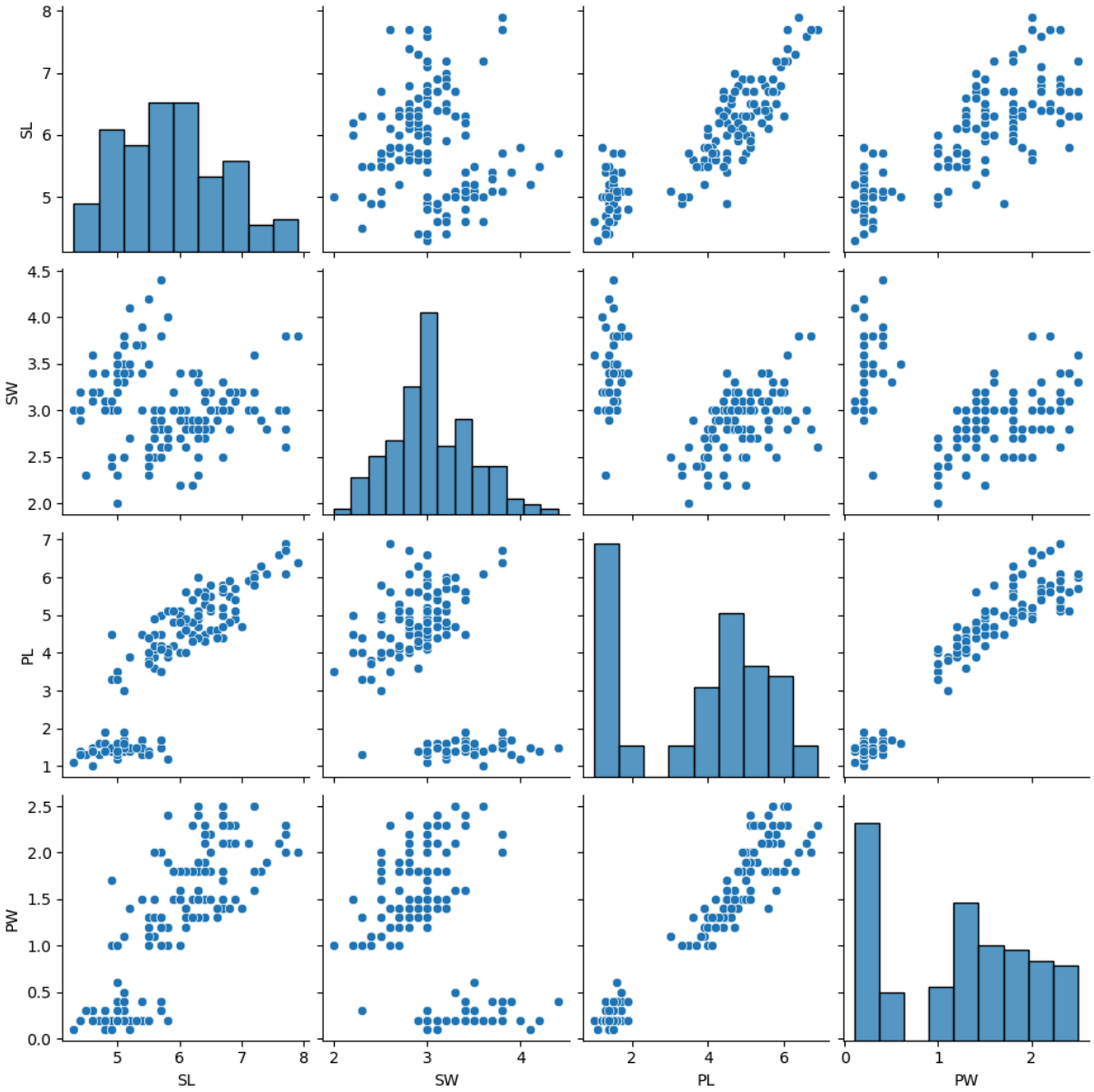
In [31]:
```
sns.kdeplot(data=iris)
plt.show()
```



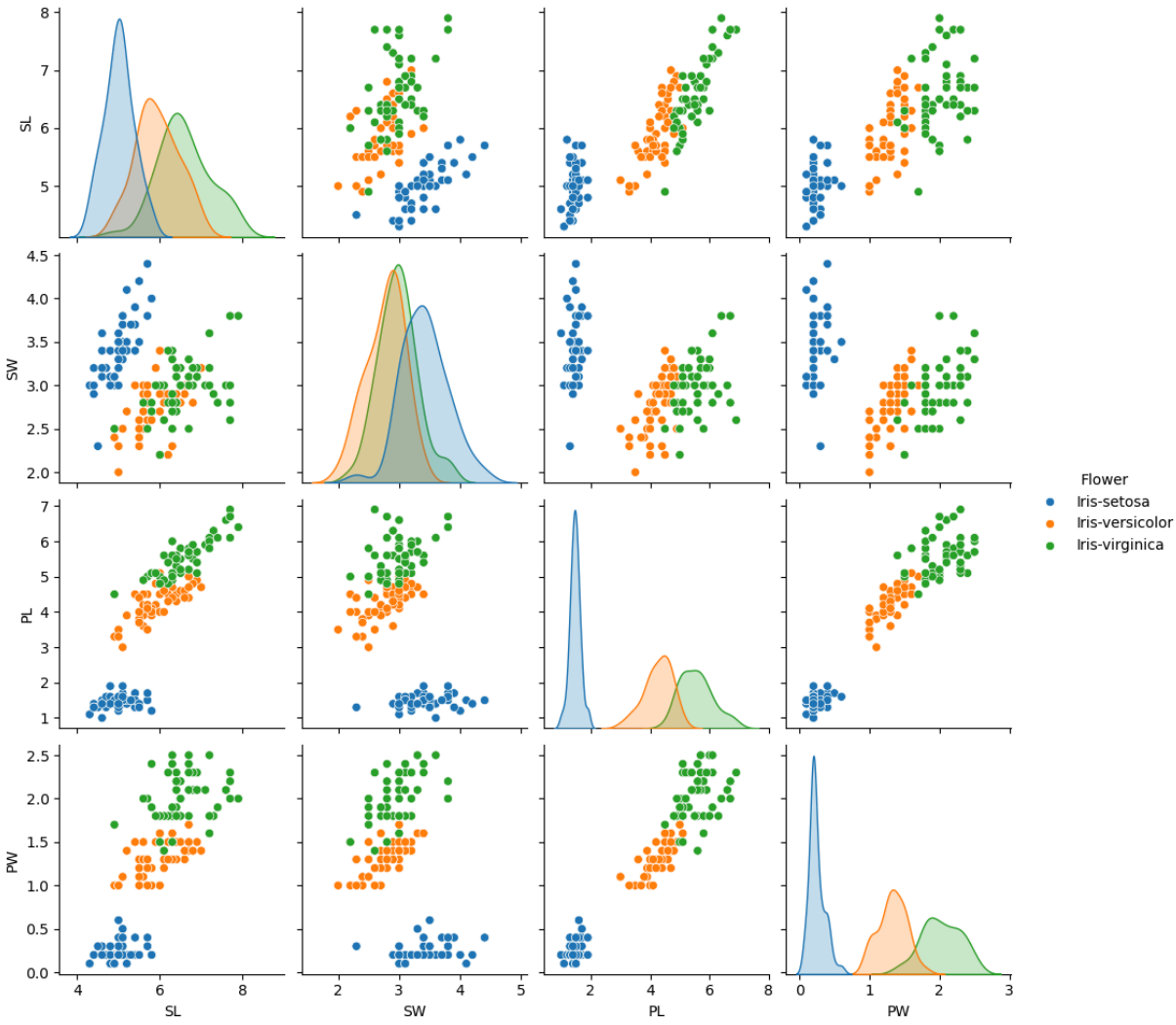
In [33]:
```
sns.kdeplot(data=iris,fill=True)
plt.show()
```



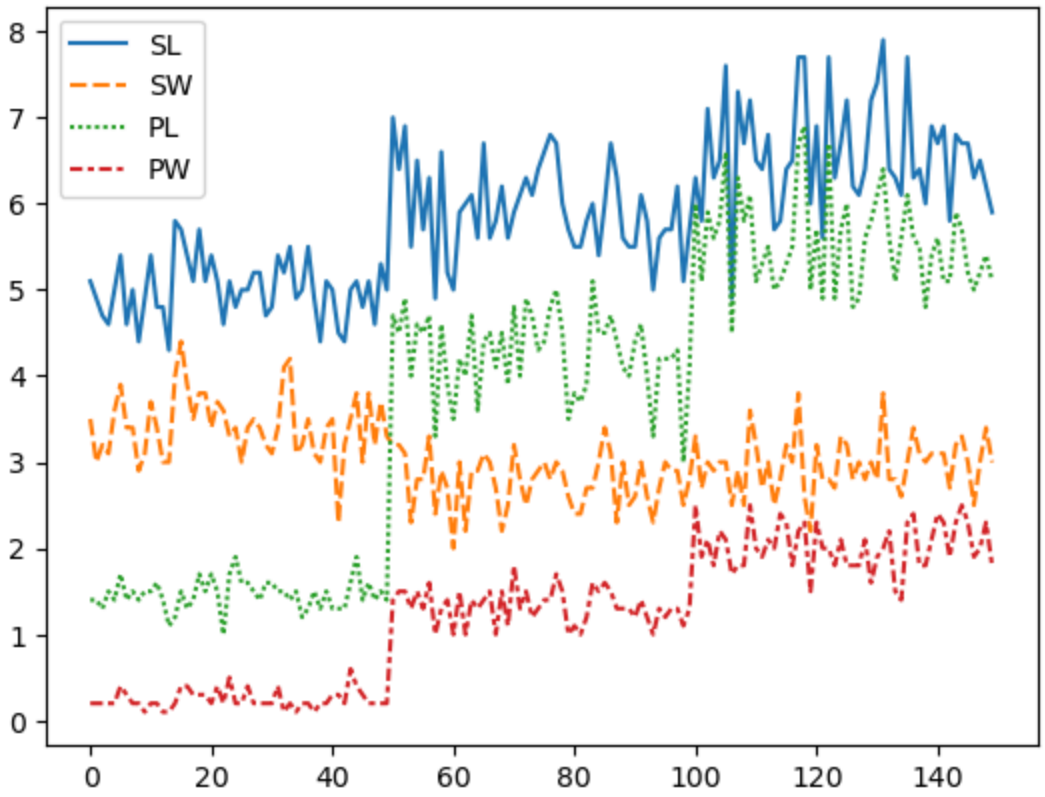
In [35]:
```
sns.pairplot(iris)
plt.show()
```



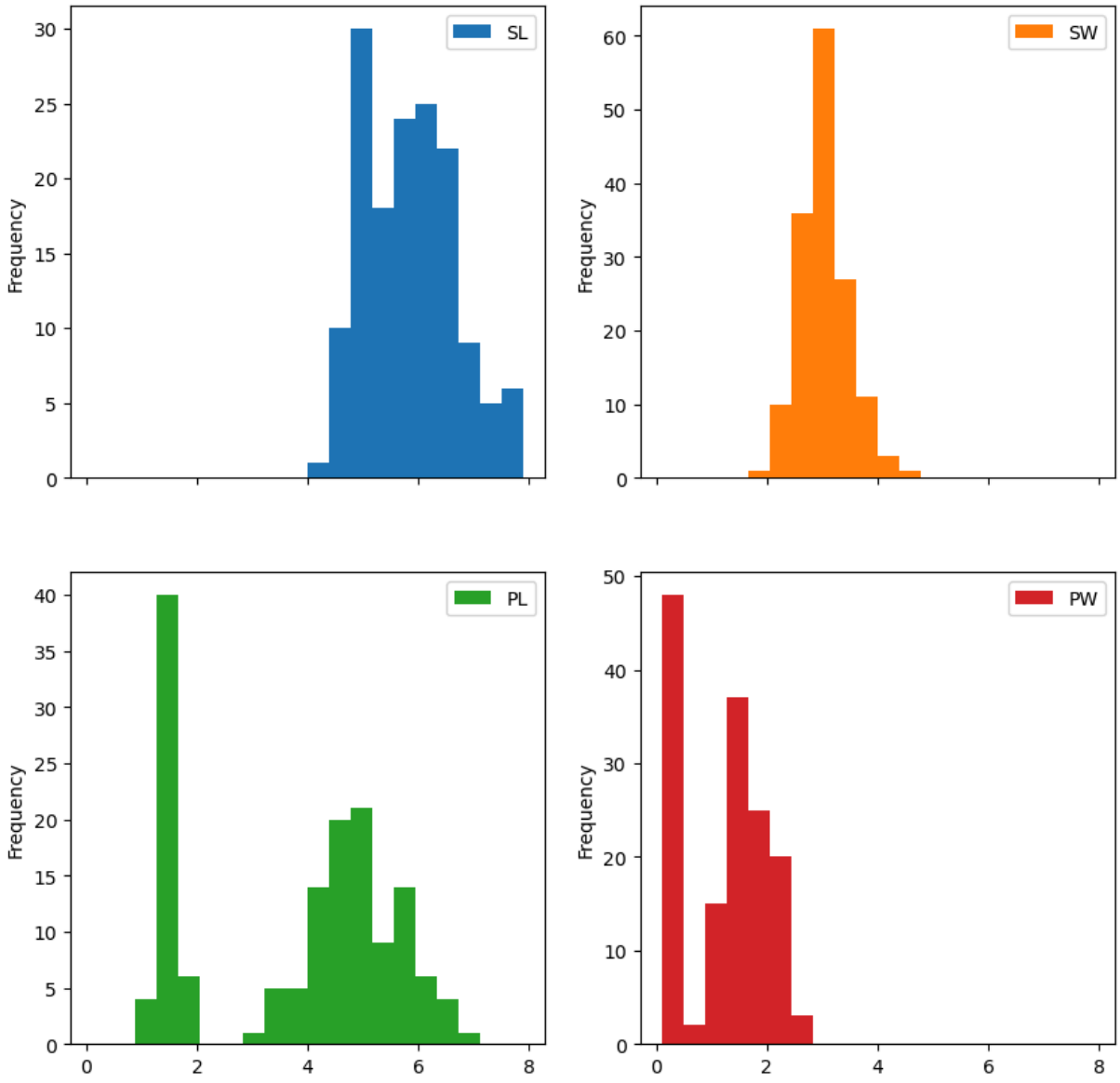
In [37]:
```
sns.pairplot(iris,hue='Flower')
plt.show()
```



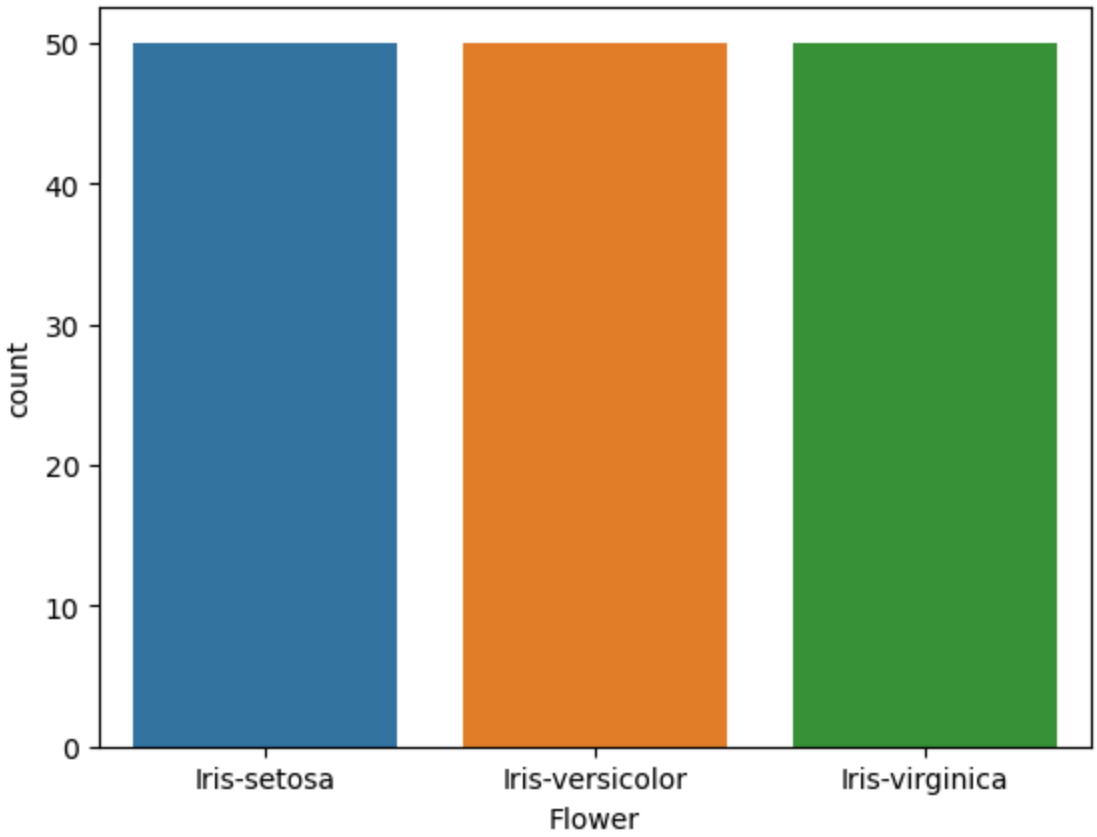
In [39]:
```
sns.lineplot(data=iris.drop(['Flower'], axis=1))
plt.show()
```



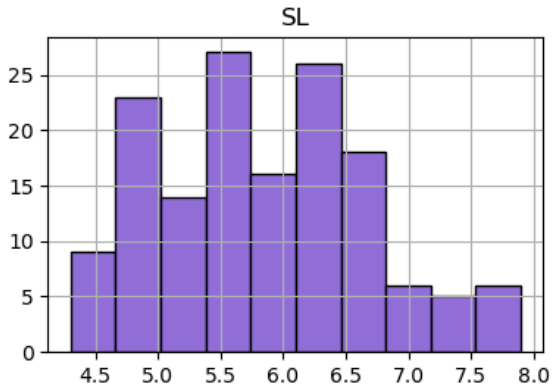
In [41]:
```
iris.plot.hist(subplots=True, layout=(2,2), figsize=(10, 10), bins=20)
plt.show()
```

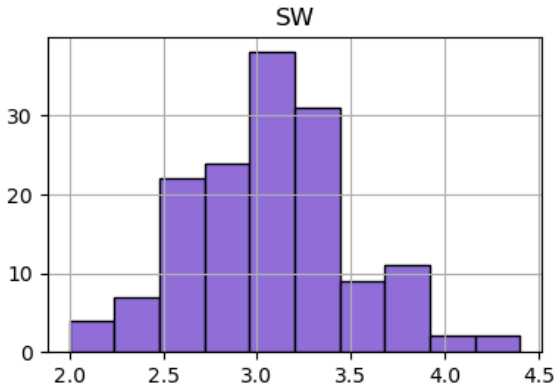


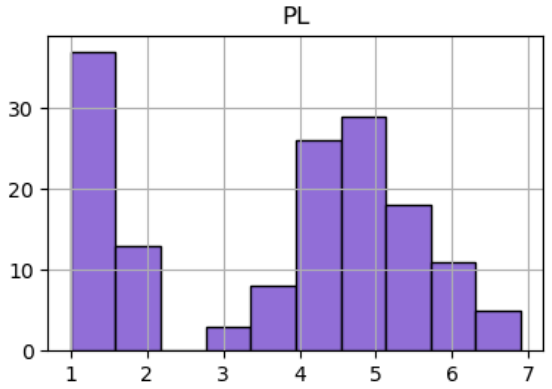
In [45]:
```
sns.countplot(x=iris.Flower,hue=iris.Flower)
plt.show()
```

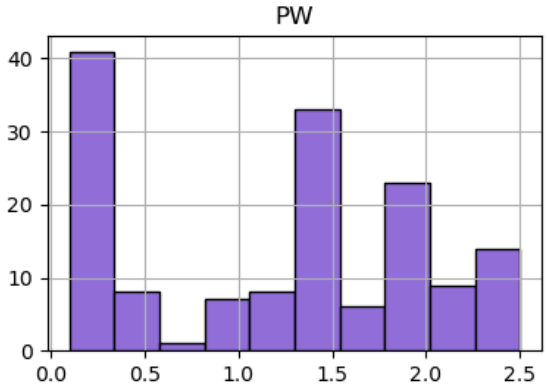


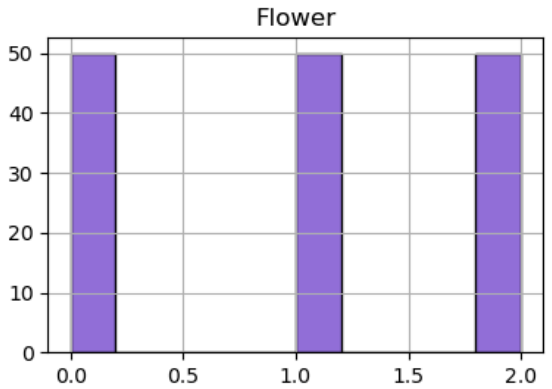
In [85]:
```
iris.hist(color= 'mediumpurple' ,edgecolor='black',figsize=(10,10))
plt.show()
```











In [47]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [49]:
```
iris.Flower = le.fit_transform(iris.Flower)
iris.Flower.unique()
```

Out[49]:
```
array([0, 1, 2])
```

In [51]:
```
le.inverse_transform([0,1,2])
```

Out[51]:
```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [53]:
```
iris.Flower = le.fit_transform(iris.Flower)
set(iris.Flower)
```

Out[53]:
```
{0, 1, 2}
```

In [55]:
```
le.inverse_transform([0,1,2])
```

Out[55]:
```
array([0, 1, 2])
```

In [59]:
```
iris.head()
```

Out[59]:

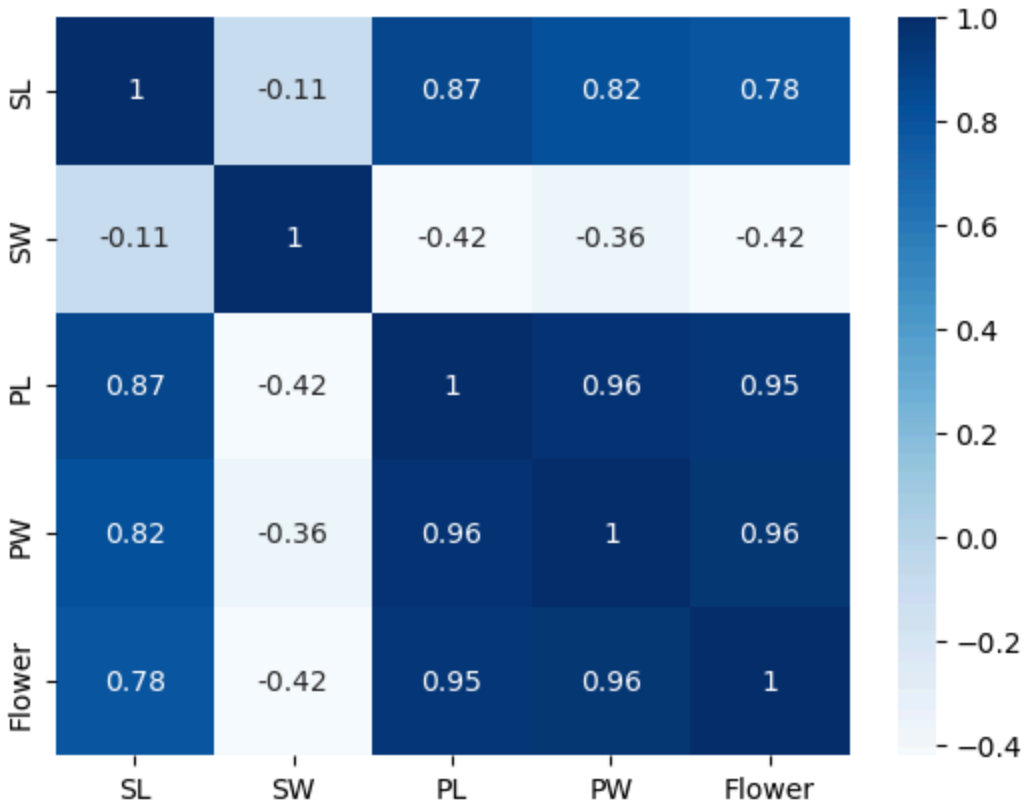
	SL	SW	PL	PW	Flower
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [61]:
```
c = iris.corr()
c
```

Out[61]:

	SL	SW	PL	PW	Flower
SL	1.000000	-0.109369	0.871754	0.817954	0.782561
SW	-0.109369	1.000000	-0.420516	-0.356544	-0.419446
PL	0.871754	-0.420516	1.000000	0.962757	0.949043
PW	0.817954	-0.356544	0.962757	1.000000	0.956464
Flower	0.782561	-0.419446	0.949043	0.956464	1.000000

In [63]:
```
sns.heatmap(c,annot=True,cmap='Blues')
plt.show()
```



In [65]:
```
x = iris.drop('Flower',axis=1)
y = iris.Flower
```

In [67]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
```

In [69]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```

In [71]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train,y_train)
```

Out[71]:
```
▾ LogisticRegression

LogisticRegression()
```

In [73]:
```
pred = lr.predict(x_test)
pred
```

Out[73]:
```
array([0, 2, 1, 2, 2, 0, 0, 2, 2, 1, 2, 1, 0, 2, 2, 2, 0, 1, 2, 0, 0, 2,
       2, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 2, 2, 2, 1, 0, 0, 2, 1,
       1], dtype=int64)
```

In [75]:
```
y_test
```

Out[75]:
```
37     0
72     1
85     1
136    2
134    2
40     0
28     0
108    2
139    2
98     1
104    2
80     1
0      0
110    2
145    2
105    2
3      0
71     1
128    2
22     0
39     0
115    2
68     1
43     0
24     0
64     1
33     0
2      0
45     0
58     1
36     0
54     1
30     0
97     1
95     1
84     1
77     1
116    2
117    2
73     1
21     0
19     0
124    2
66     1
88     1
Name: Flower, dtype: int64
```

In [77]:
```
from sklearn.metrics import accuracy_score,recall_score,precision_score,f1_score

ac = accuracy_score(pred,y_test)
print('Accuracy : ',ac)
re = recall_score(pred,y_test,average='weighted')
pr = precision_score(pred,y_test,average='weighted')
f1 = f1_score(pred,y_test,average='weighted')

print('Recall : ',re)
```

```
print('Precision : ',pr)
print('F1 Score : ',f1)
```

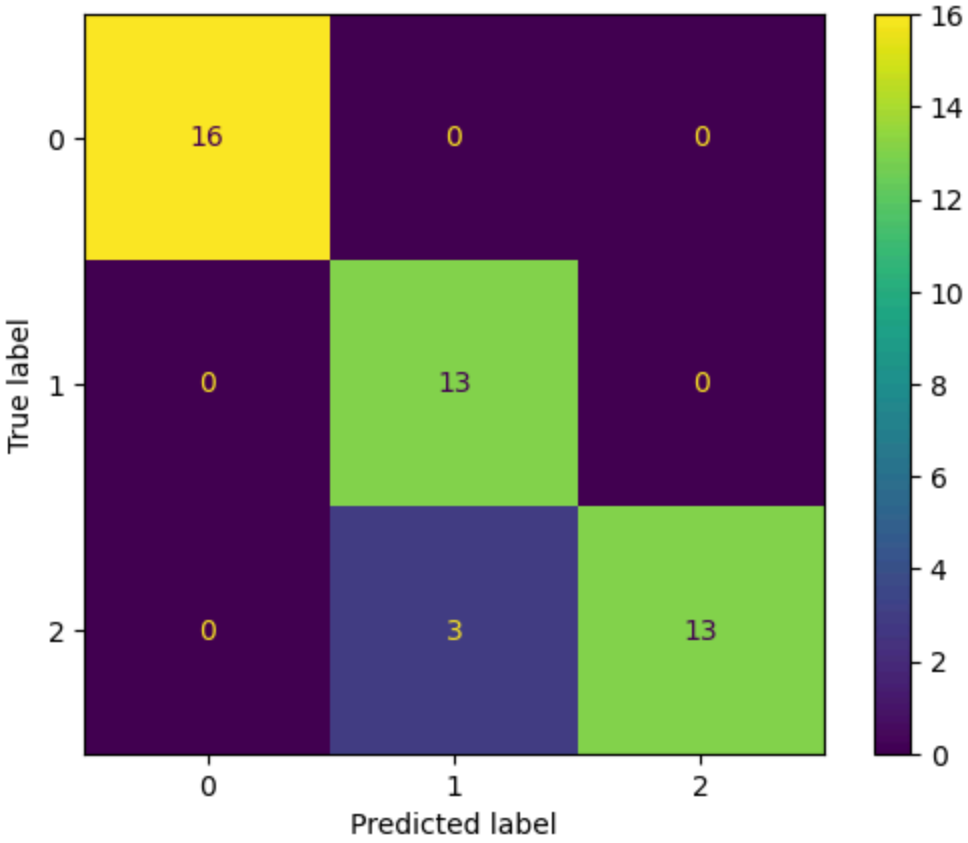
```
Accuracy :  0.9333333333333333
Recall :  0.9333333333333333
Precision :  0.9458333333333333
F1 Score :  0.9333333333333333
```

In [79]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(pred,y_test)
cm
```

Out[79]:
```
array([[16,  0,  0],
       [ 0, 13,  0],
       [ 0,  3, 13]], dtype=int64)
```

In [81]:
```
cmd = ConfusionMatrixDisplay(cm)
cmd.plot()
plt.show()
```



In [87]:
```
from sklearn.svm import SVC
svm = SVC(kernel='rbf', random_state=0, gamma=.10, C=1.0)
svm.fit(x_train, y_train)

svm.score(x_test, y_test)
```

Out[87]:
```
0.9333333333333333
```

In [89]:
```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(x_train, y_train)

dtree.score(x_test, y_test)
```

Out[89]:
```
0.9333333333333333
```

In [91]:
```
df = pd.DataFrame({'Y_Test':list(y_test),'Prediction':pred})
df
```

Out[91]:

	Y_Test	Prediction
0	0	0
1	1	2
2	1	1
3	2	2
4	2	2
5	0	0
6	0	0
7	2	2
8	2	2
9	1	1
10	2	2
11	1	1
12	0	0
13	2	2
14	2	2
15	2	2
16	0	0
17	1	1
18	2	2
19	0	0
20	0	0
21	2	2
22	1	2
23	0	0
24	0	0
25	1	1
26	0	0
27	0	0
28	0	0
29	1	1

	Y_Test	Prediction
30	0	0
31	1	1
32	0	0
33	1	1
34	1	1
35	1	1
36	1	2
37	2	2
38	2	2
39	1	1
40	0	0
41	0	0
42	2	2
43	1	1
44	1	1