IRIS FLOWERS CLASSIFICATION ML PROJECT

Import Libraries

In [42]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import LabelEncoder, MinMaxScaler,StandardScaler
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import seaborn as sns
```

Load the Data

In [43]:
```
df=pd.read_csv(r'C:\Users\user\Downloads\iris.csv')
df
```

Out[43]:

	Unnamed: 0	sepal_length	sepal_width	petal_length	petal_width	species
0	0	5.1	3.5	1.4	0.2	setosa
1	1	4.9	3.0	1.4	0.2	setosa
2	2	4.7	3.2	1.3	0.2	setosa
3	3	4.6	3.1	1.5	0.2	setosa
4	4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...	...
145	145	6.7	3.0	5.2	2.3	virginica
146	146	6.3	2.5	5.0	1.9	virginica
147	147	6.5	3.0	5.2	2.0	virginica
148	148	6.2	3.4	5.4	2.3	virginica
149	149	5.9	3.0	5.1	1.8	virginica

150 rows × 6 columns

Basic Chacks

In [44]:
```
df.head()
```

Loading [MathJax]/extensions/Safe.js

Out[44]:

	Unnamed: 0	sepal_length	sepal_width	petal_length	petal_width	species
0	0	5.1	3.5	1.4	0.2	setosa
1	1	4.9	3.0	1.4	0.2	setosa
2	2	4.7	3.2	1.3	0.2	setosa
3	3	4.6	3.1	1.5	0.2	setosa
4	4	5.0	3.6	1.4	0.2	setosa

In [45]:
```
df.tail()
```

Out[45]:

	Unnamed: 0	sepal_length	sepal_width	petal_length	petal_width	species
145	145	6.7	3.0	5.2	2.3	virginica
146	146	6.3	2.5	5.0	1.9	virginica
147	147	6.5	3.0	5.2	2.0	virginica
148	148	6.2	3.4	5.4	2.3	virginica
149	149	5.9	3.0	5.1	1.8	virginica

In [46]:
```
df.describe()
```

Out[46]:

	Unnamed: 0	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	74.500000	5.843333	3.057333	3.758000	1.199333
std	43.445368	0.828066	0.435866	1.765298	0.762238
min	0.000000	4.300000	2.000000	1.000000	0.100000
25%	37.250000	5.100000	2.800000	1.600000	0.300000
50%	74.500000	5.800000	3.000000	4.350000	1.300000
75%	111.750000	6.400000	3.300000	5.100000	1.800000
max	149.000000	7.900000	4.400000	6.900000	2.500000

In [47]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Unnamed: 0    150 non-null    int64  
 1   sepal_length  150 non-null    float64
 2   sepal_width   150 non-null    float64
 3   petal_length  150 non-null    float64
 4   petal_width   150 non-null    float64
 5   species       150 non-null    object 
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

In [49]:
```
df.drop('Unnamed: 0',axis=1,inplace=True)
```

In [50]:
```
df.dtypes
```

Loading [MathJax]/extensions/Safe.js

Out[50]:
```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

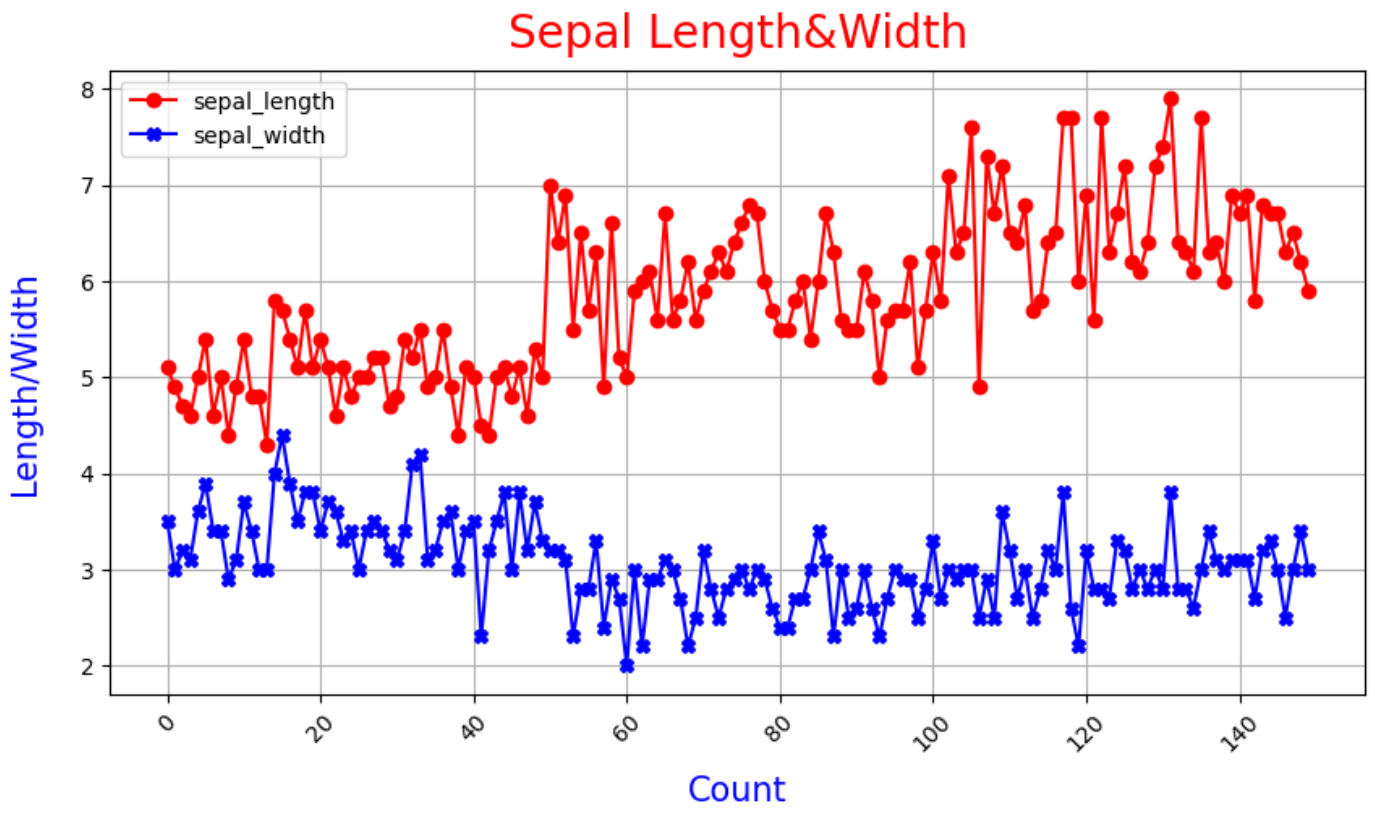
In [51]:
```
df['species']=df['species'].str.replace('Iris-','')
df['species']
```

Out[51]:
```
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
          ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

Sepal length and width

In [52]:
```
df['sepal_length'].plot(kind='line',legend=True,marker='o',color='r',figsize=(10,5),grid
df['sepal_width'].plot(kind='line',legend=True,marker="X",color='b',figsize=(10,5),grid=
plt.title('Sepal Length&Width',pad=10,loc='center',fontdict={'fontsize': 20, 'color': 'r
plt.xlabel('Count',labelpad=20,loc='center',fontdict={'fontsize': 15, 'color': 'b','vert
plt.ylabel('Length/Width',labelpad=20,loc='center',fontdict={'fontsize': 15, 'color': 'b
```

Out[52]:
```
Text(0, 0.5, 'Length/Width')
```

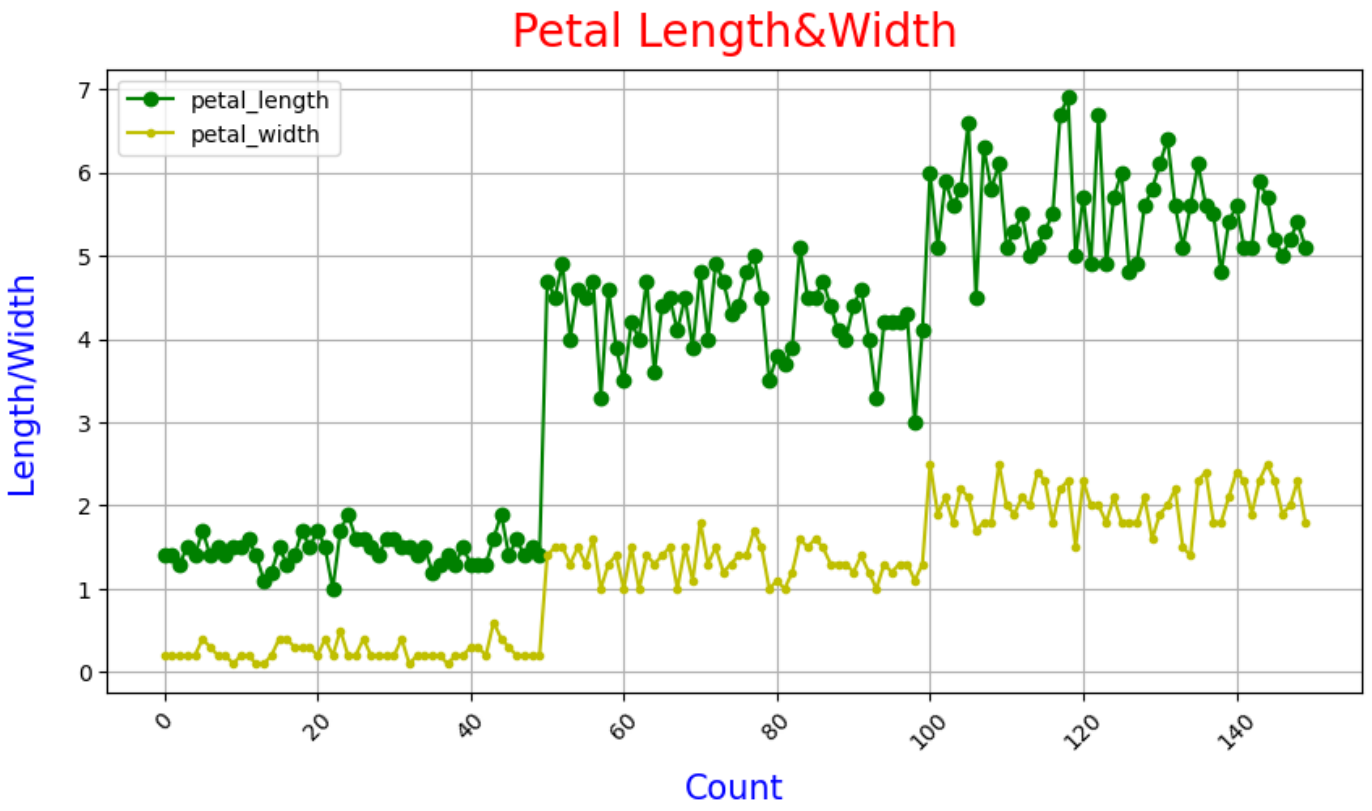


Sepal width is lesser then sepal length

Loading [MathJax]/extensions/Safe.js

In [53]:
```
df['petal_length'].plot(kind='line',legend=True,marker='o',color='g',figsize=(10,5),grid
df['petal_width'].plot(kind='line',legend=True,marker=".",color='y',figsize=(10,5),grid=
plt.title('Petal Length&Width',pad=10,loc='center',fontdict={'fontsize': 20, 'color': 'r
plt.xlabel('Count',labelpad=20,loc='center',fontdict={'fontsize': 15, 'color': 'b','vert
plt.ylabel('Length/Width',labelpad=20,loc='center',fontdict={'fontsize': 15, 'color': 'b
```

Out[53]:
```
Text(0, 0.5, 'Length/Width')
```

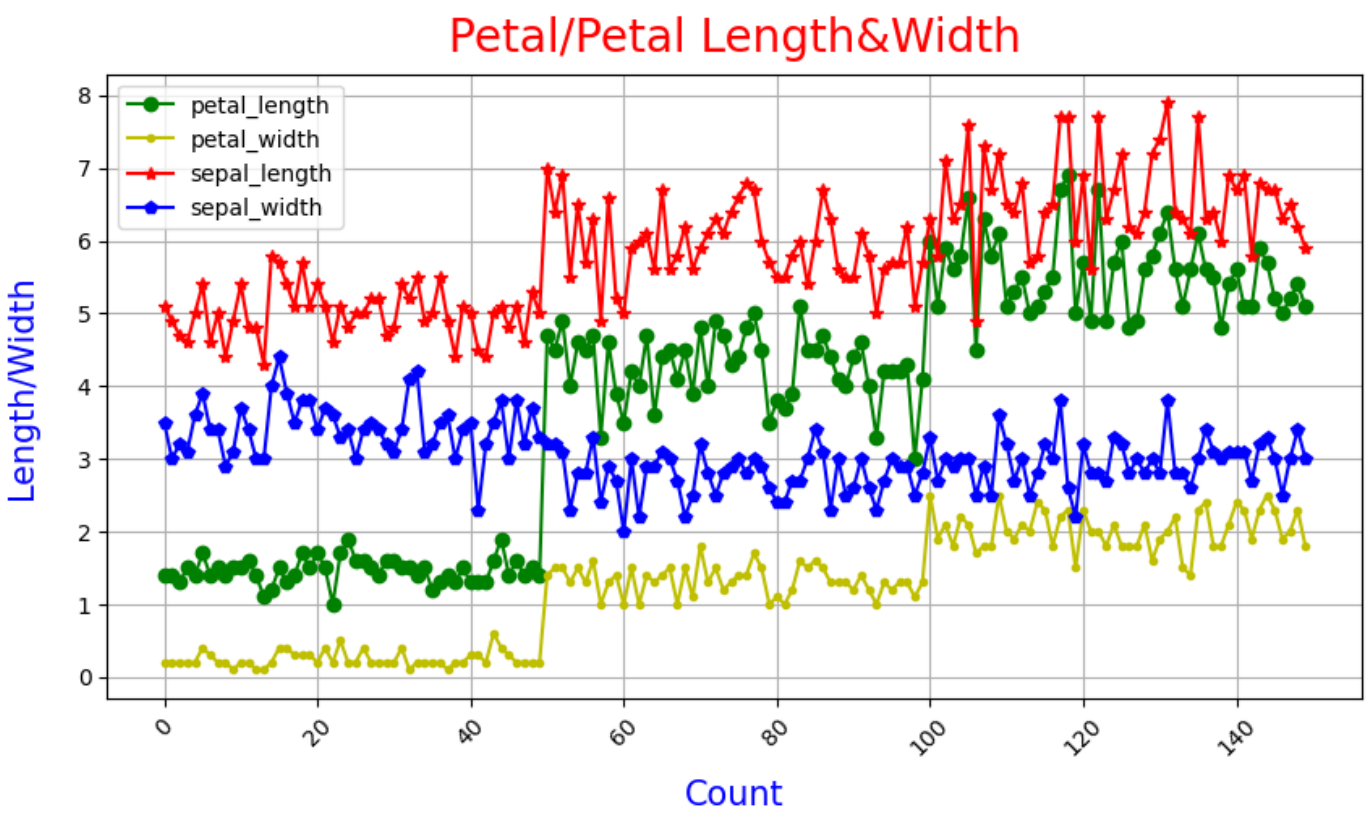


Petal length is greater then petal width

In [54]:
```
df['petal_length'].plot(kind='line',legend=True,marker='o',color='g',figsize=(10,5),grid
df['petal_width'].plot(kind='line',legend=True,marker=".",color='y',figsize=(10,5),grid=
df['sepal_length'].plot(kind='line',legend=True,marker="*",color='r',figsize=(10,5),grid
df['sepal_width'].plot(kind='line',legend=True,marker="p",color='b',figsize=(10,5),grid=
plt.title('Petal/Petal Length&Width',pad=10,loc='center',fontdict={'fontsize': 20, 'colo
plt.xlabel('Count',labelpad=20,loc='center',fontdict={'fontsize': 15, 'color': 'b','vert
plt.ylabel('Length/Width',labelpad=20,loc='center',fontdict={'fontsize': 15, 'color': 'b
```

Out[54]:
```
Text(0, 0.5, 'Length/Width')
```

Loading [MathJax]/extensions/Safe.js

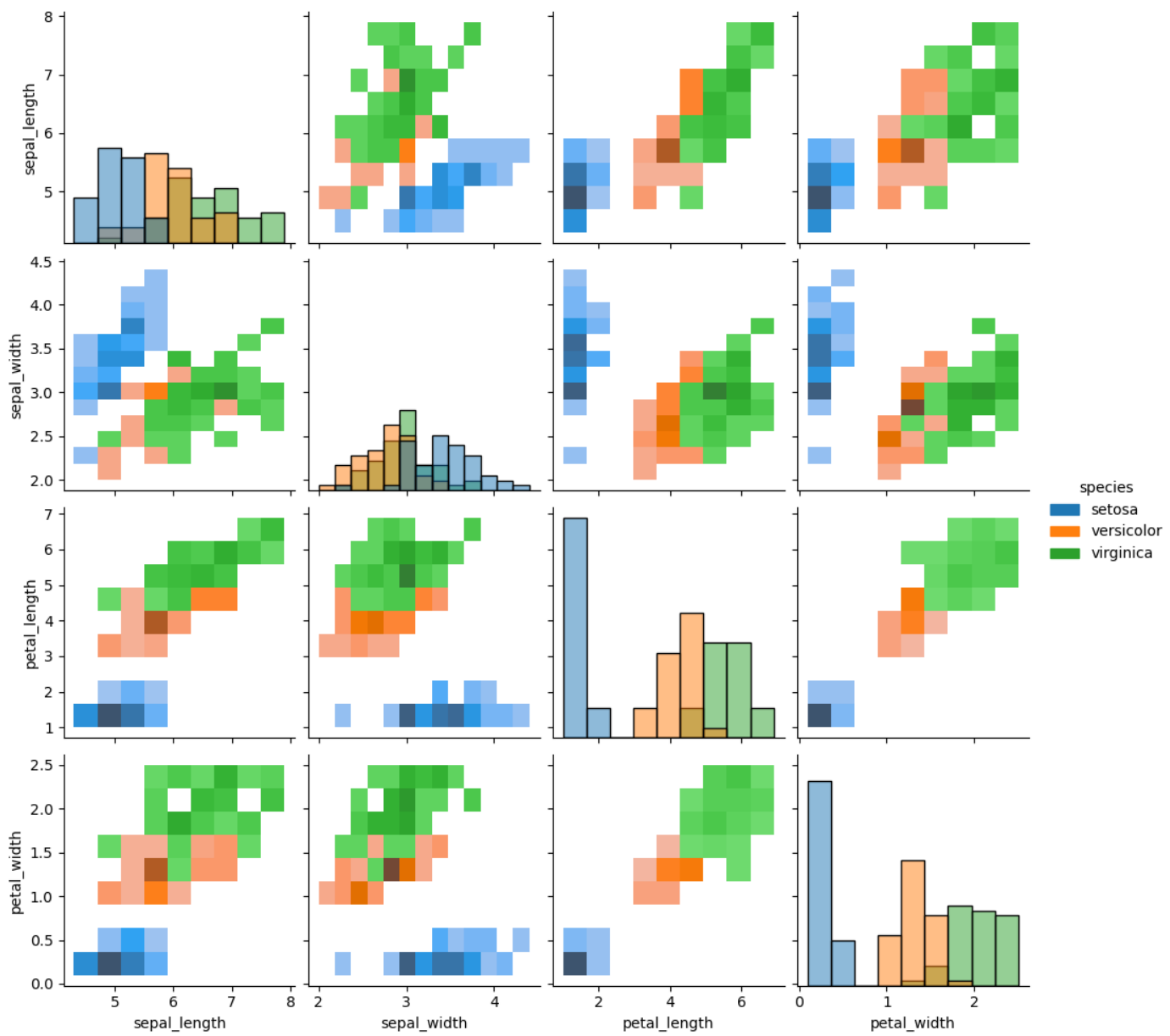


Visualization of Data

In [55]:
```
plt.figure(figsize=(17,6))
sns.pairplot(df,hue='species',kind='hist')
```

Out[55]:
```
<seaborn.axisgrid.PairGrid at 0x1e4efecc610>

<Figure size 1700x600 with 0 Axes>
```

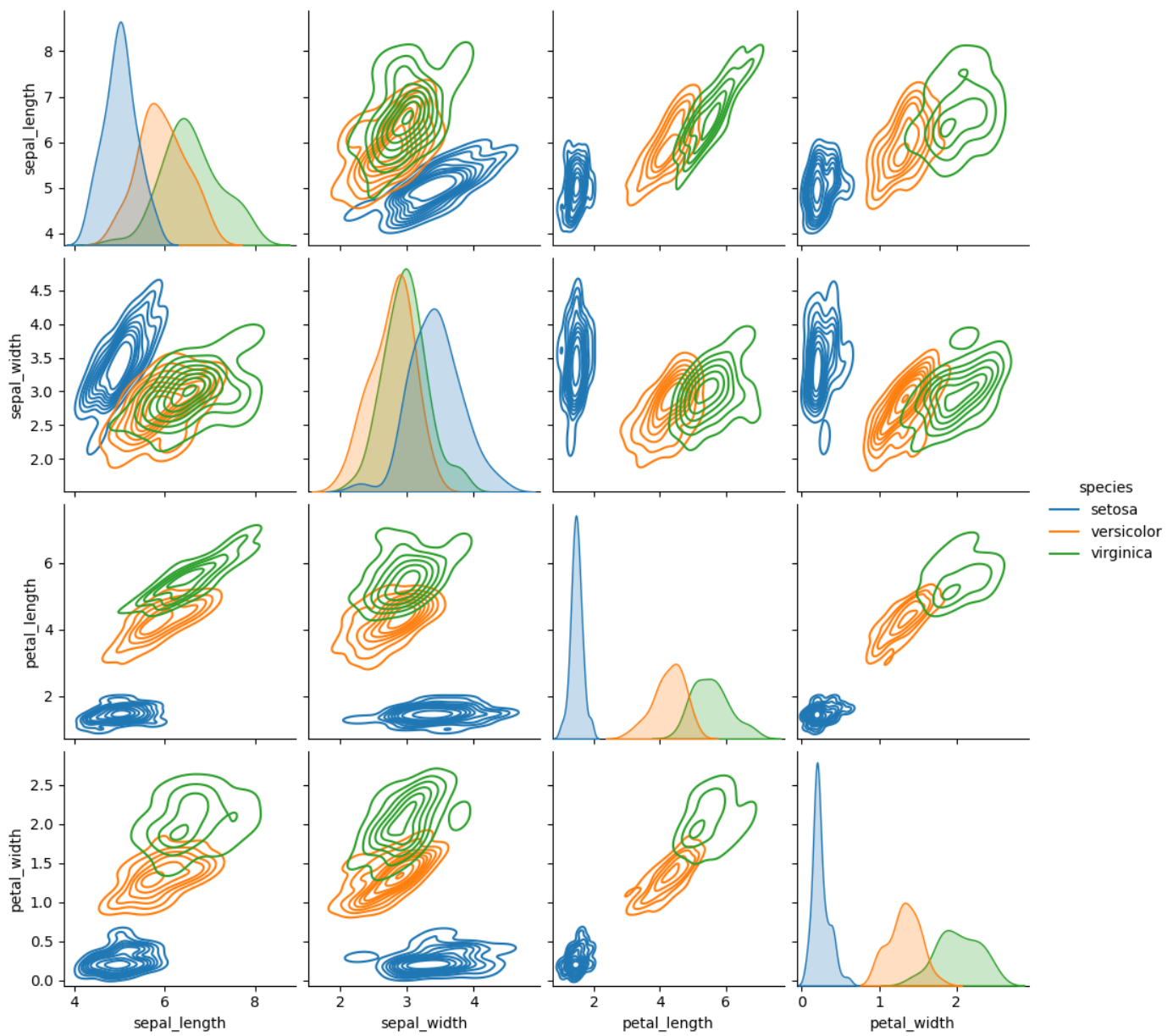
Loading [MathJax]/extensions/Safe.js



In [56]:
```
plt.figure(figsize=(17,6))
sns.pairplot(df,hue='species',kind='kde')
```

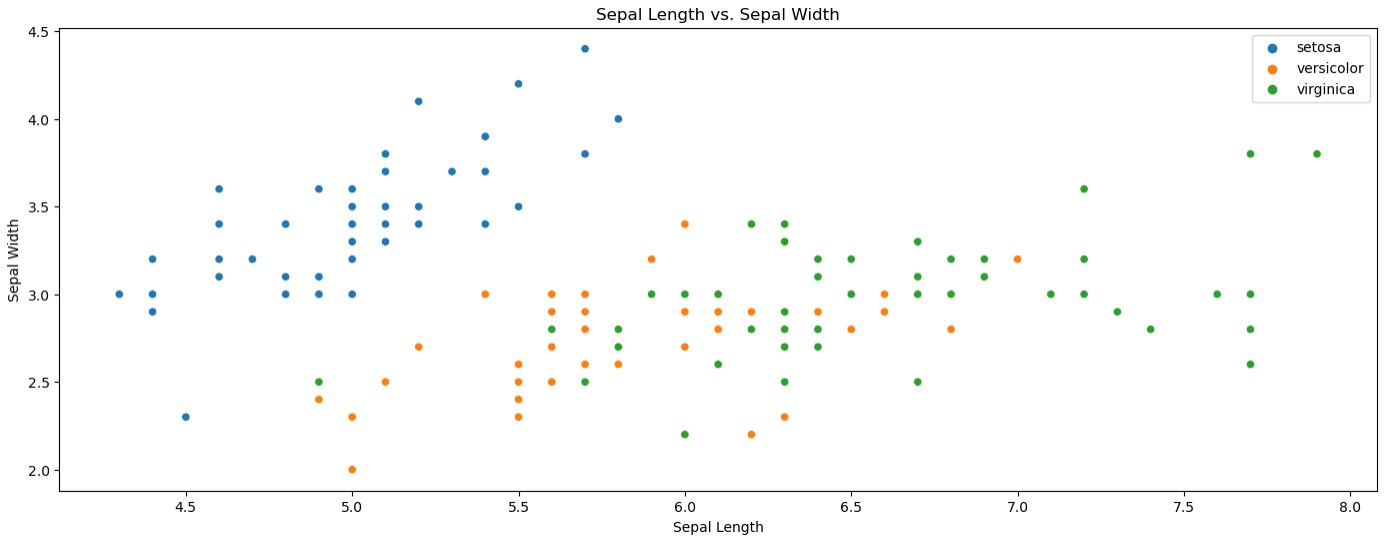
Out[56]:
```
<seaborn.axisgrid.PairGrid at 0x1e4f0883280>

<Figure size 1700x600 with 0 Axes>
```

Loading [MathJax]/extensions/Safe.js



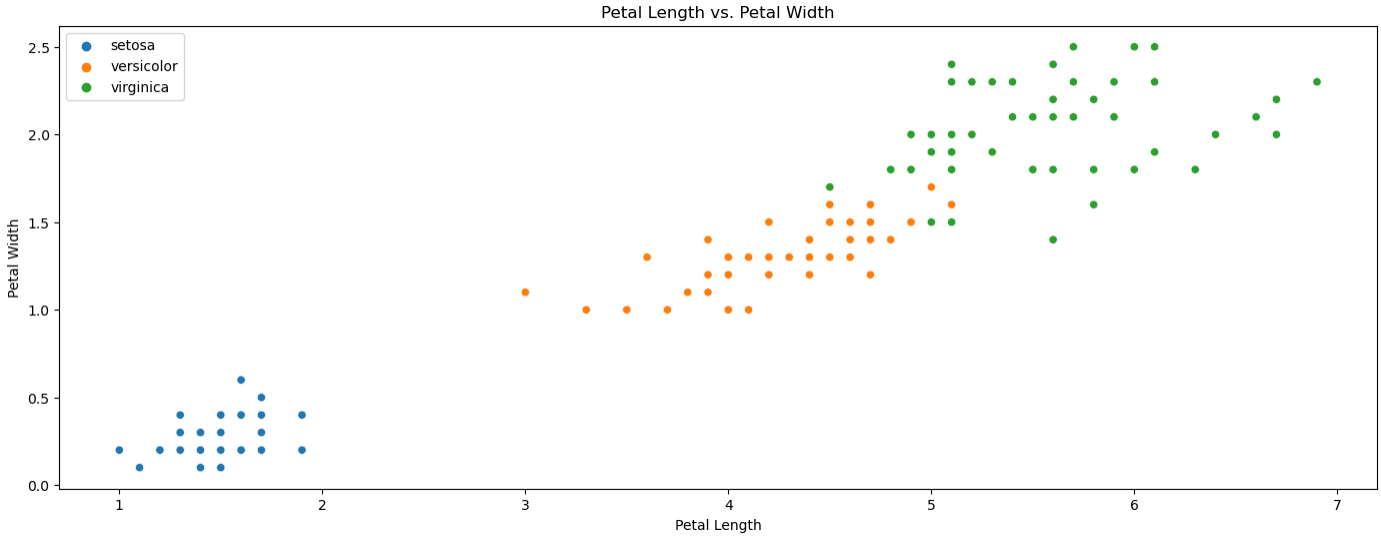
In [57]:

```
plt.figure(figsize=(17, 6))
sns.scatterplot(data=df, x='sepal_length', y='sepal_width', hue='species')
plt.legend(loc='upper right')
plt.title('Sepal Length vs. Sepal Width')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.show()
```

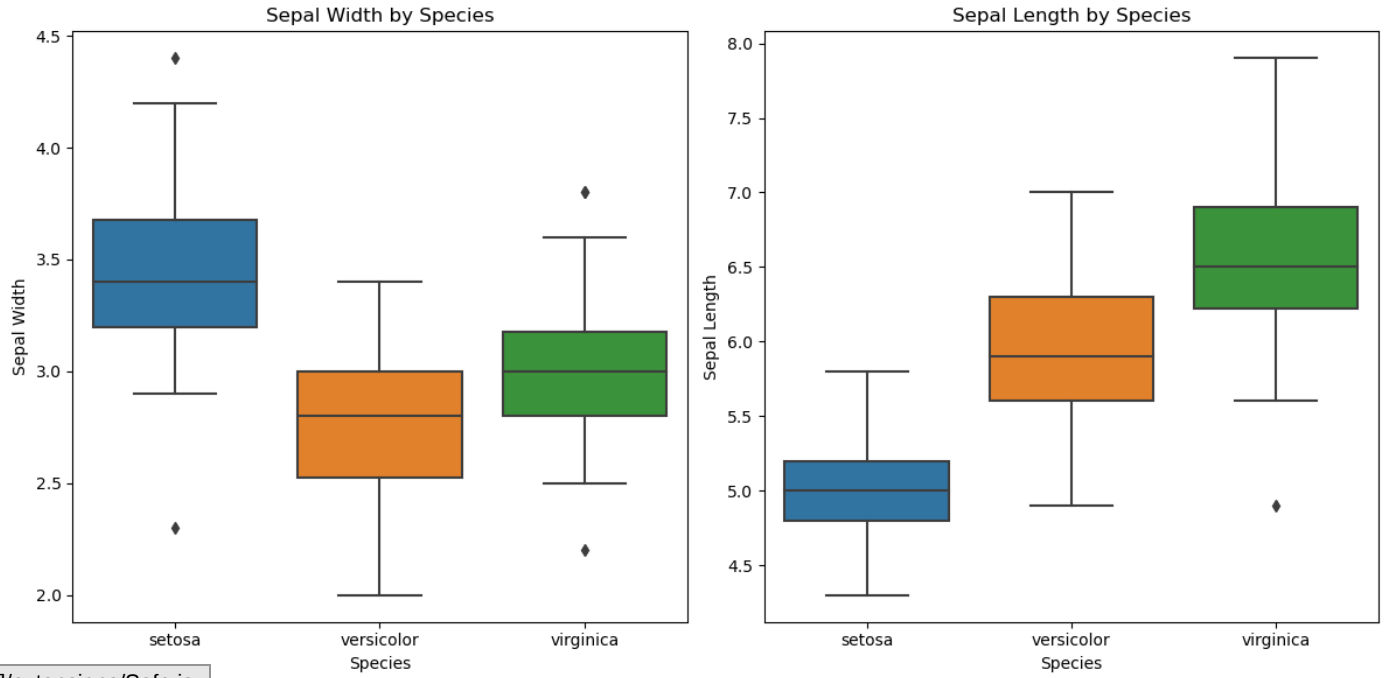


Loading [MathJax]/extensions/Safe.js

In [58]:
```
plt.figure(figsize=(17, 6))
sns.scatterplot(data=df, x='petal_length', y='petal_width', hue='species')
plt.legend(loc='upper left')
plt.title('Petal Length vs. Petal Width')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.show()
```



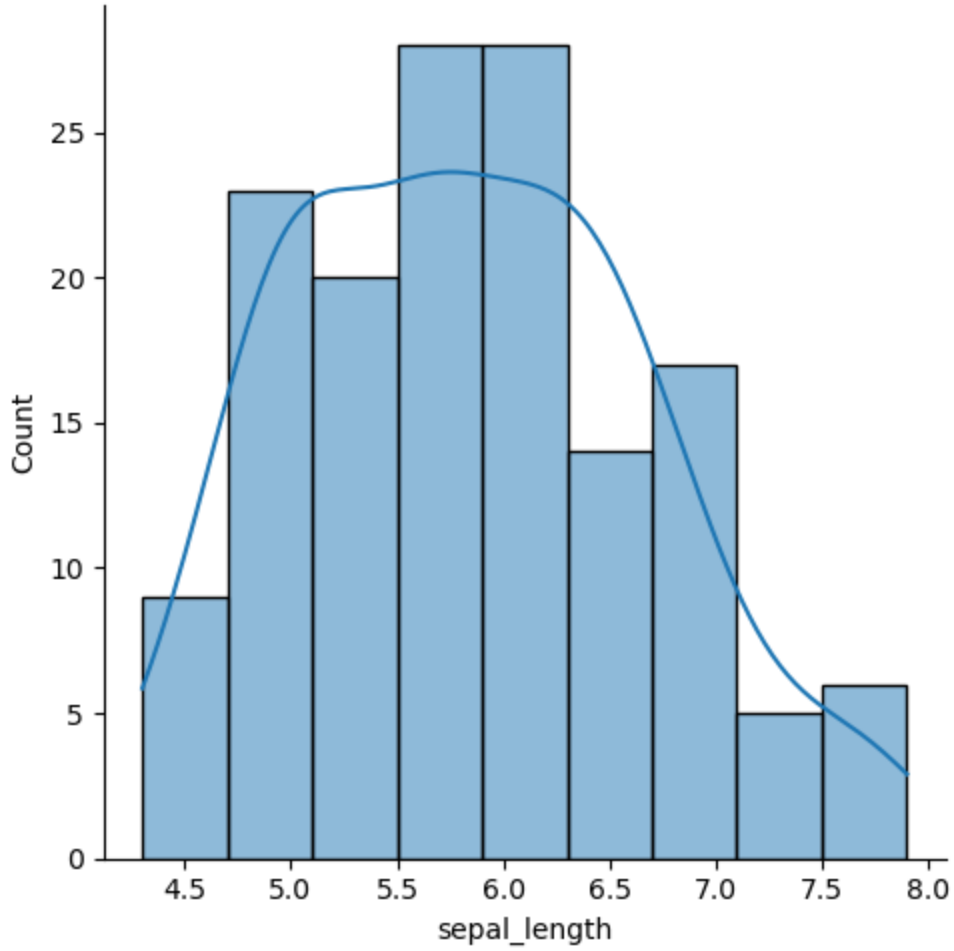
In [59]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 6))

# Box plot for sepal width
sns.boxplot(data=df, x='species', y='sepal_width', ax=axes[0])
axes[0].set_title('Sepal Width by Species')
axes[0].set_xlabel('Species')
axes[0].set_ylabel('Sepal Width')

# Box plot for sepal length
sns.boxplot(data=df, x='species', y='sepal_length', ax=axes[1])
axes[1].set_title('Sepal Length by Species')
axes[1].set_xlabel('Species')
axes[1].set_ylabel('Sepal Length')

plt.tight_layout()
plt.show()
```



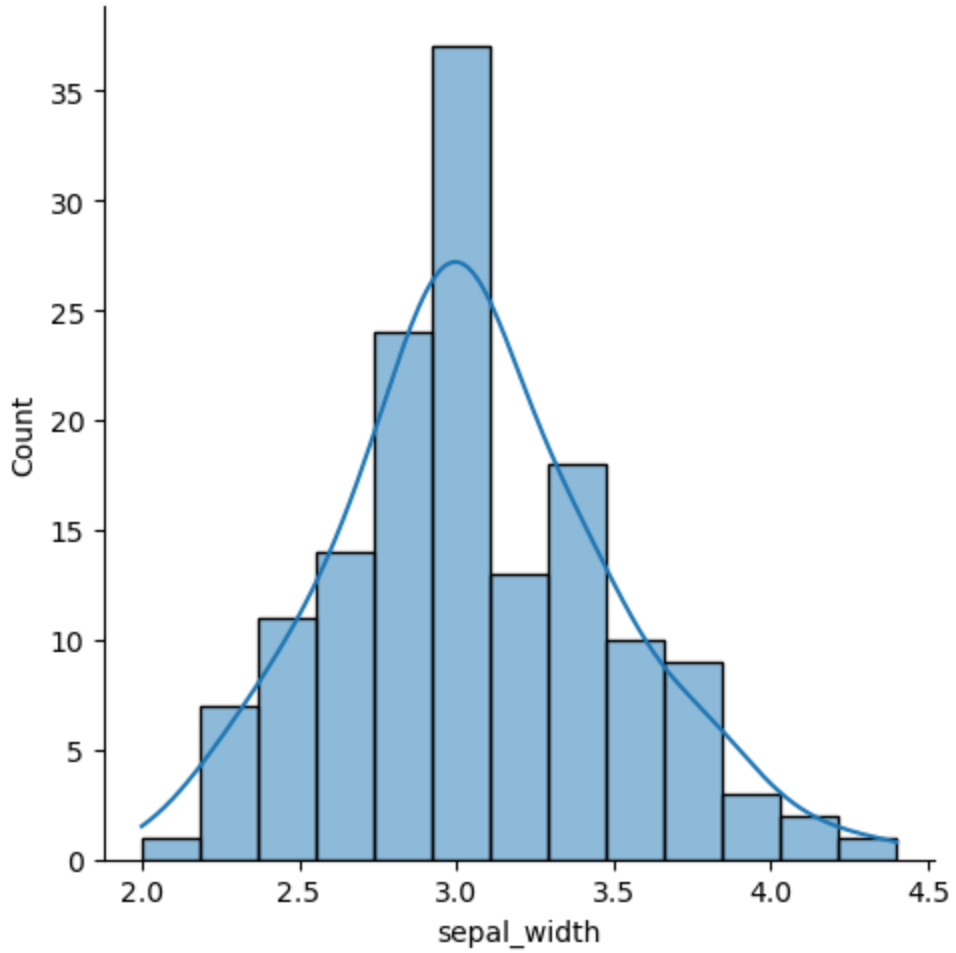
Loading [MathJax]/extensions/Safe.js

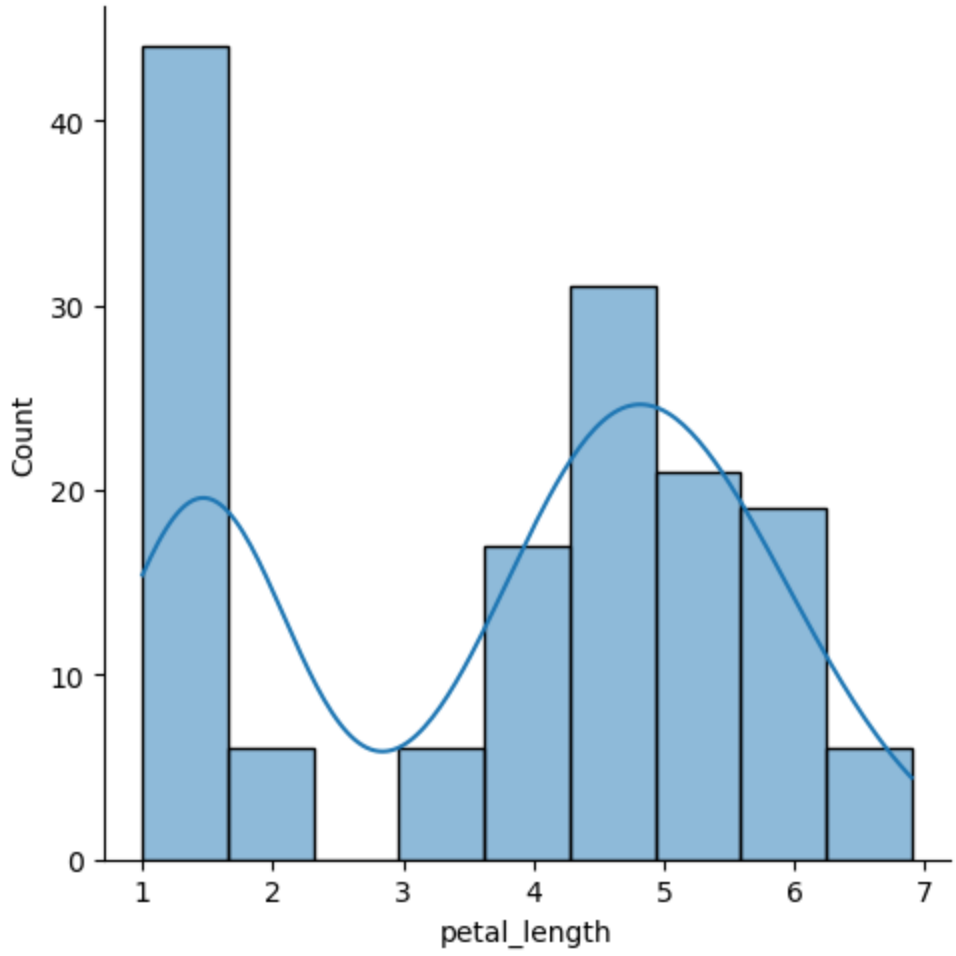
In [60]:
```
plt.figure(figsize=(10, 6))

sns.displot(data=df, x='sepal_length', label='Sepal Length', kde=True)
sns.displot(data=df, x='sepal_width', label='Sepal Width', kde=True)
sns.displot(data=df, x='petal_length', label='Petal Length', kde=True)
sns.displot(data=df, x='petal_width', label='Petal Width', kde=True)

plt.legend()
plt.title('Density Distribution of Iris Flower Features')
plt.xlabel('Feature Values')
plt.ylabel('Density')
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```

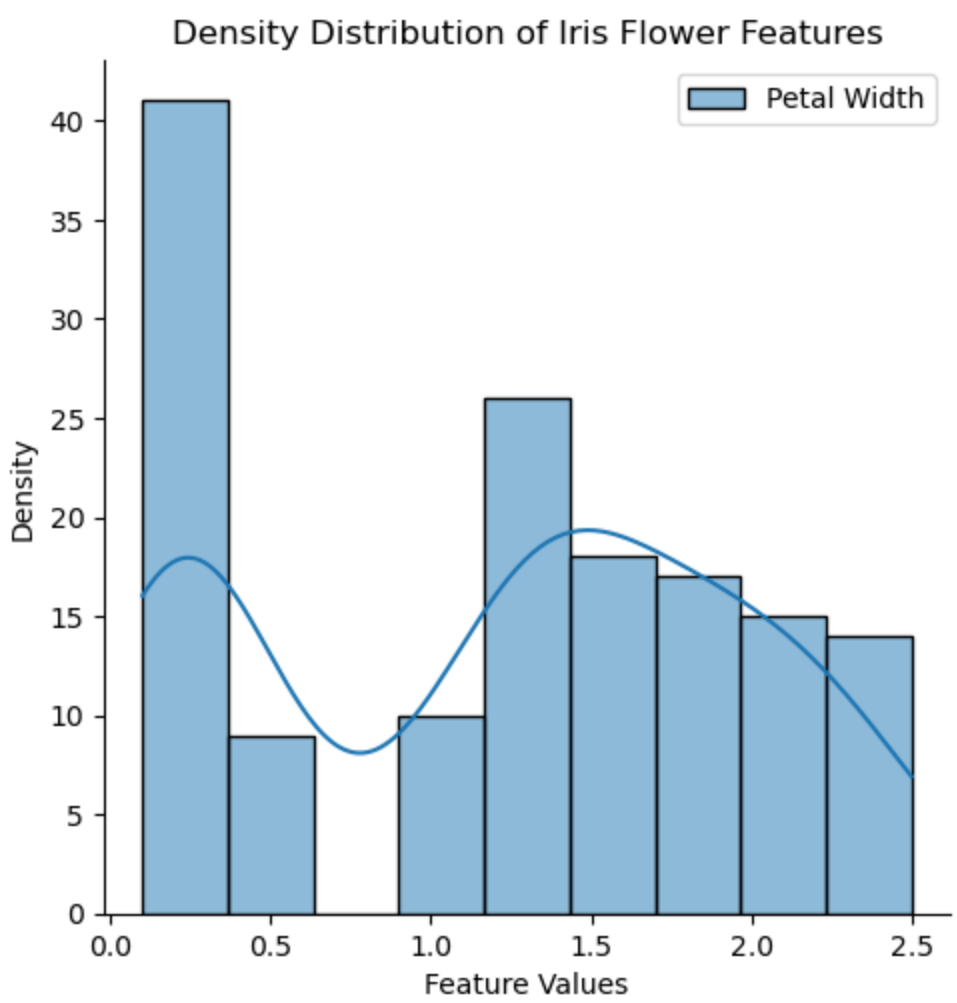


Loading [MathJax]/extensions/Safe.js





Loading [MathJax]/extensions/Safe.js



In [61]:
```
le = LabelEncoder()
```

In [62]:
```
df['species']=le.fit_transform(df['species'])
df
```

Out[62]:

	sepal_length	sepal_width	petal_length	petal_width	species
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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

Model Train and Test

Loading [MathJax]/extensions/Safe.js

In [63]:
```
X_train,X_test,y_train,y_test = train_test_split(df.iloc[:,:4],df['species'],test_size=0
```

In [64]:
```
lor = LogisticRegression()
```

In [65]:
```
lor.fit(X_train,y_train)
```

Out[65]:
```
▼ LogisticRegression
LogisticRegression()
```

In [66]:
```
test = lor.predict(X_test)
for tests in test:
    print("->",tests)
```

```
-> 0
-> 0
-> 2
-> 0
-> 0
-> 2
-> 0
-> 2
-> 2
-> 0
-> 0
-> 0
-> 0
-> 0
-> 1
-> 1
-> 0
-> 1
-> 2
-> 1
-> 2
-> 1
-> 2
-> 1
-> 1
-> 0
-> 0
-> 2
-> 0
-> 2
```

In [67]:
```
lor.score(X_test,y_test)
```

Out[67]:
```
0.9666666666666667
```

In [68]:
```
y_test.values
```

Out[68]:
```
array([0, 0, 2, 0, 0, 2, 0, 2, 2, 0, 0, 0, 0, 0, 1, 1, 0, 1, 2, 1, 1, 1,
       2, 1, 1, 0, 0, 2, 0, 2])
```

Using feature scaling

In [69]:
```
minmax = MinMaxScaler()
```

Loading [MathJax]/extensions/Safe.js

In [70]:
```
x_train= minmax.fit_transform(X_train)
x_test= minmax.fit_transform(X_test)
```

In [71]:
```
lor.fit(x_train,y_train)
```

Out[71]:
```
▼ LogisticRegression

LogisticRegression()
```

In [72]:
```
test1= lor.predict(x_test)
for test2 in test1:
    print("->",test2)
```

```
-> 0
-> 0
-> 2
-> 0
-> 0
-> 2
-> 0
-> 2
-> 2
-> 0
-> 0
-> 0
-> 0
-> 0
-> 2
-> 2
-> 0
-> 1
-> 2
-> 1
-> 2
-> 1
-> 2
-> 1
-> 1
-> 0
-> 0
-> 2
-> 0
-> 2
```

In [73]:
```
lor.score(x_test,y_test)
```

Out[73]:
```
0.9
```

In [ ]:

Loading [MathJax]/extensions/Safe.js