Iris Flowers Classification

Importing the required Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import seaborn as sns
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import silhouette_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

Read the Dataset

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

Exploratory Data Analysis on the Dataset

In [3]:
```
iris.head()
```

Out[3]:

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

In [4]:
```
iris.tail()
```

Out[4]:

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

In [5]:
```
iris.shape
```

Out[5]:(150, 5)

In [6]:
```
iris.columns
```

Out[6]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

In [7]:
```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [8]:
```
iris.describe()
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width
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 [9]:
```
iris.isnull().sum()
```

Out[9]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [10]:
```
iris.isnull()
```

Out[10]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

150 rows × 5 columns

In [11]:
```
class_counts = iris['species'].value_counts()
print("Number of classes:", len(class_counts))
print("\nClass names and their counts:")
print(class_counts)
```

```
Number of classes: 3

Class names and their counts:
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

In [12]:
```
versicolor_subset = iris[iris['species'] == 'versicolor'].iloc[:5]
print("\nVersicolor subset:")
print(versicolor_subset)
```

```
Versicolor subset:
    sepal_length  sepal_width  petal_length  petal_width     species
50           7.0          3.2           4.7          1.4  versicolor
51           6.4          3.2           4.5          1.5  versicolor
52           6.9          3.1           4.9          1.5  versicolor
53           5.5          2.3           4.0          1.3  versicolor
54           6.5          2.8           4.6          1.5  versicolor
```

In [13]:
```
virginica_subset = iris[iris['species'] == 'virginica'].iloc[:5]
print("\nVirginica subset:")
print(virginica_subset)
```

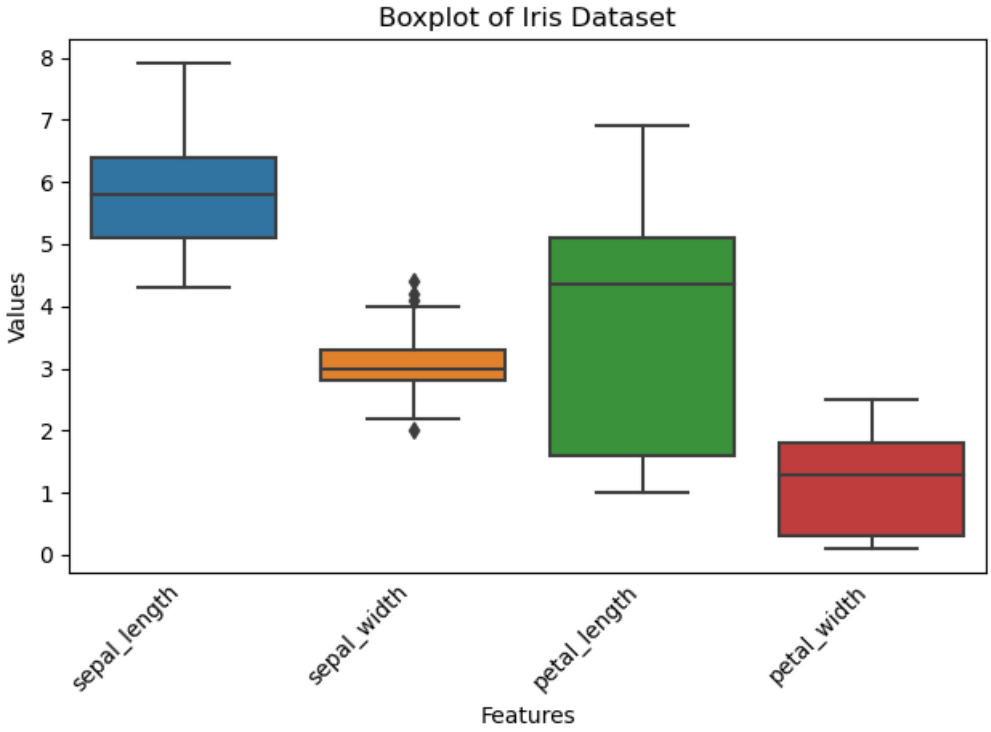
```
Virginica subset:
     sepal_length  sepal_width  petal_length  petal_width    species
100           6.3          3.3           6.0          2.5  virginica
101           5.8          2.7           5.1          1.9  virginica
102           7.1          3.0           5.9          2.1  virginica
103           6.3          2.9           5.6          1.8  virginica
104           6.5          3.0           5.8          2.2  virginica
```

In [14]:
```
petal_length_less_than_2 = iris[iris['petal_length'] < 2]
print(petal_length_less_than_2['species'].value_counts())
```

setosa 50
Name: species, dtype: int64
Data Visualization

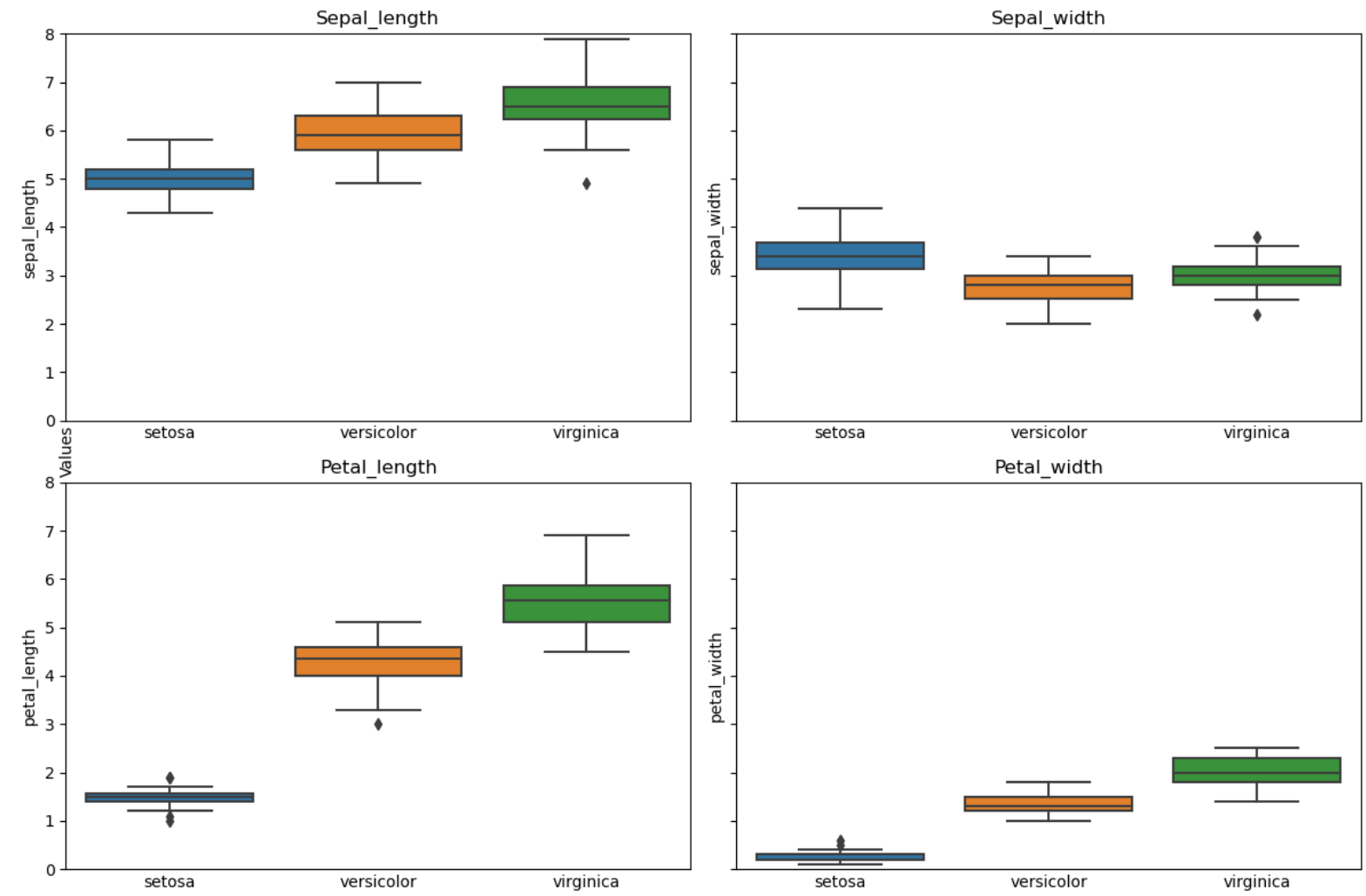
Box Plots

In [37]:
```
sns.boxplot(data=iris)

plt.tick_params(axis='x', which='both', length=0)
plt.xticks(rotation=45, ha='right')


plt.xlabel('Features')
plt.ylabel('Values')
plt.title('Boxplot of Iris Dataset')


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



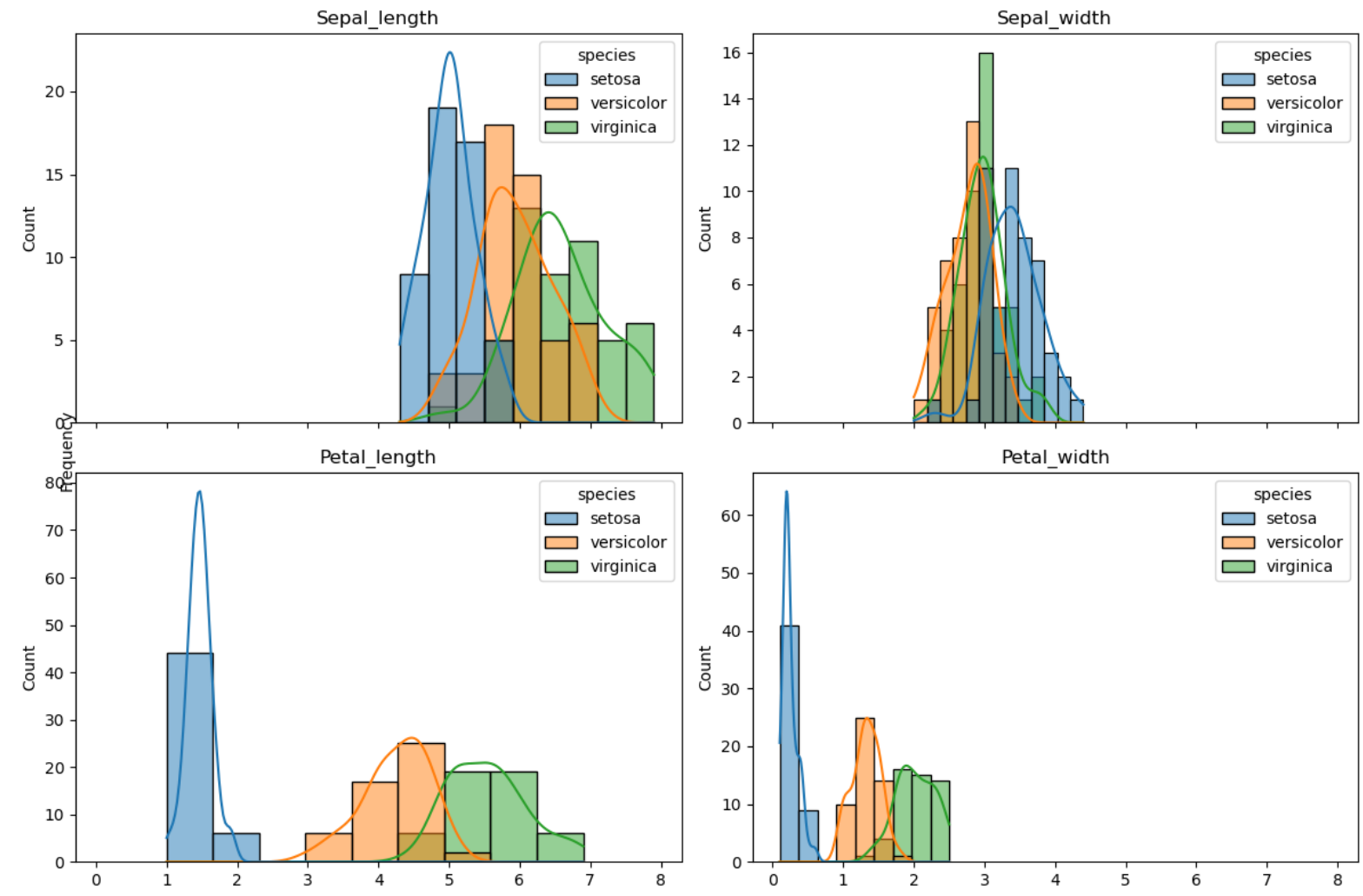
Sub Plot

In [38]:
```
fig, axes = plt.subplots(2, 2, figsize=(12, 8), sharey=True)

axes = axes.flatten()

features = iris.drop('species', axis=1)
for i, feature in enumerate(features):
    sns.boxplot(x='species', y=feature, data=iris, ax=axes[i])
    axes[i].set_title(feature.capitalize())
    axes[i].set_ylim(0, 8)
    axes[i].tick_params(axis='x', which='both', length=0)
    axes[i].set_xlabel('')

for ax in axes[len(features):]:
    ax.remove()

fig.text(0.04, 0.5, 'Values', va='center', rotation='vertical')
plt.tight_layout()

plt.show()
```



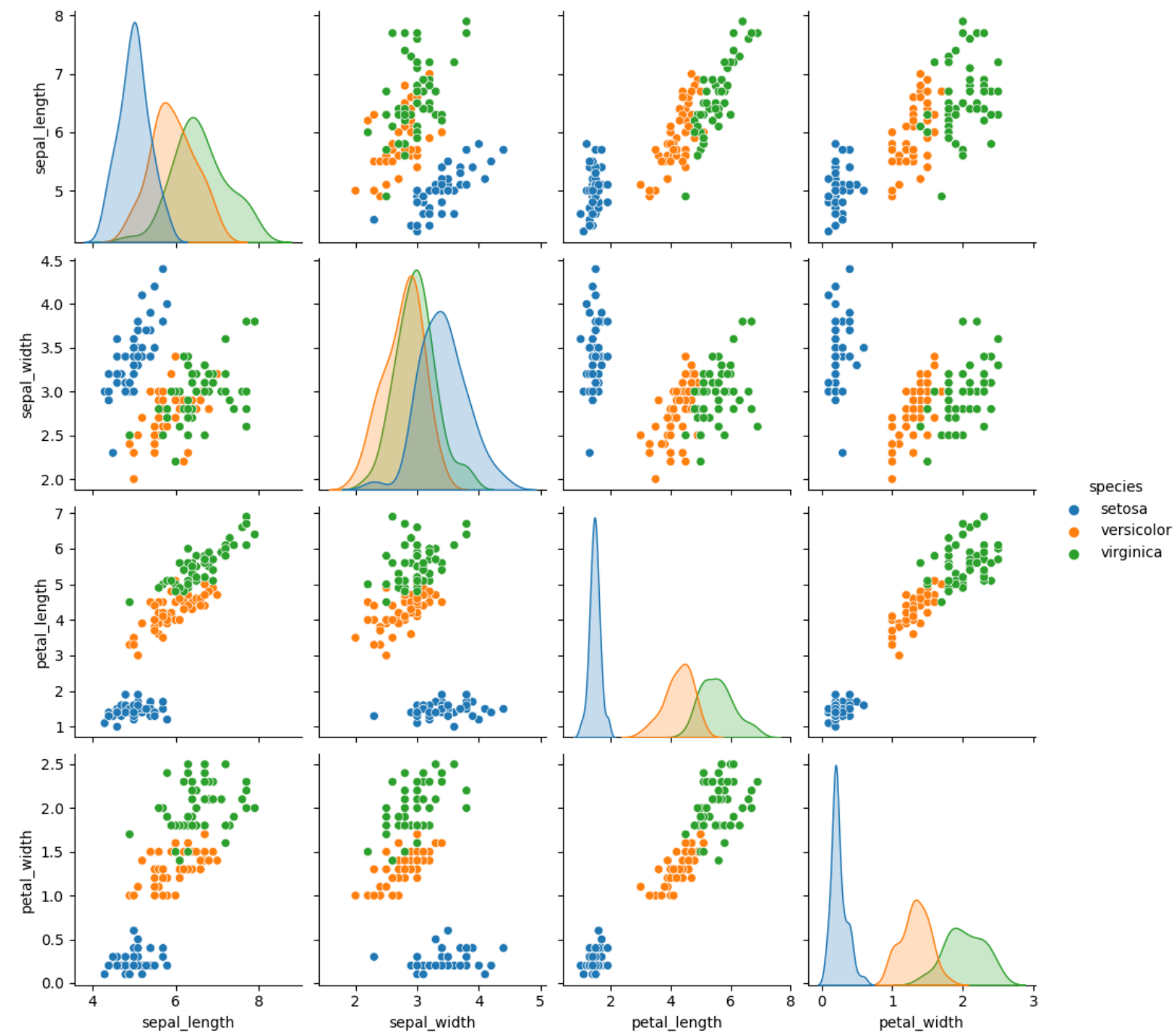
Scatter Plot

In [39]:
```
fig, axes = plt.subplots(2, 2, figsize=(12, 8), sharex=True)
axes = axes.flatten()
for i, feature in enumerate(features):
    sns.histplot(data=iris, x=feature, hue='species', ax=axes[i], kde=True)
    axes[i].set_title(feature.capitalize())
    axes[i].set_xlabel('')

fig.text(0.04, 0.5, 'Frequency', va='center', rotation='vertical')
plt.tight_layout()

plt.show()
```



Pair Plot

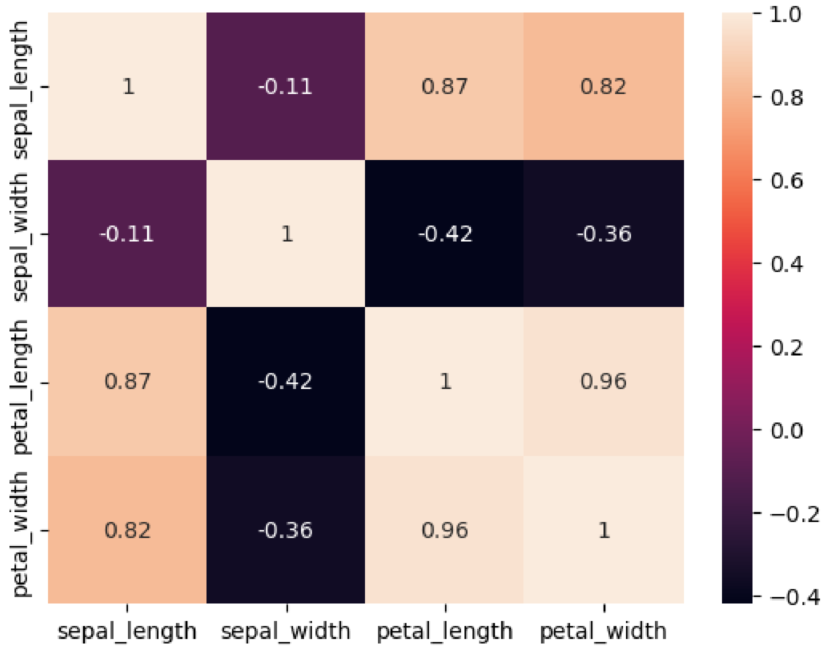
In [40]:
```
sns.pairplot(data=iris, hue='species')
plt.show()
```



Heat Maps

In [41]:
```
correl=iris.corr()
sns.heatmap(correl,annot=True)
```

C:\Users\A.Surya Tejaswini\AppData\Local\Temp\ipykernel_30648\434761437.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  correl=iris.corr()
Out[41]:<Axes: >



Data Processing

In [42]:
```
target = iris['species']
```

In [43]:
```
scaler = StandardScaler()
```

In [44]:
```
features_scaled = scaler.fit_transform(features)
```

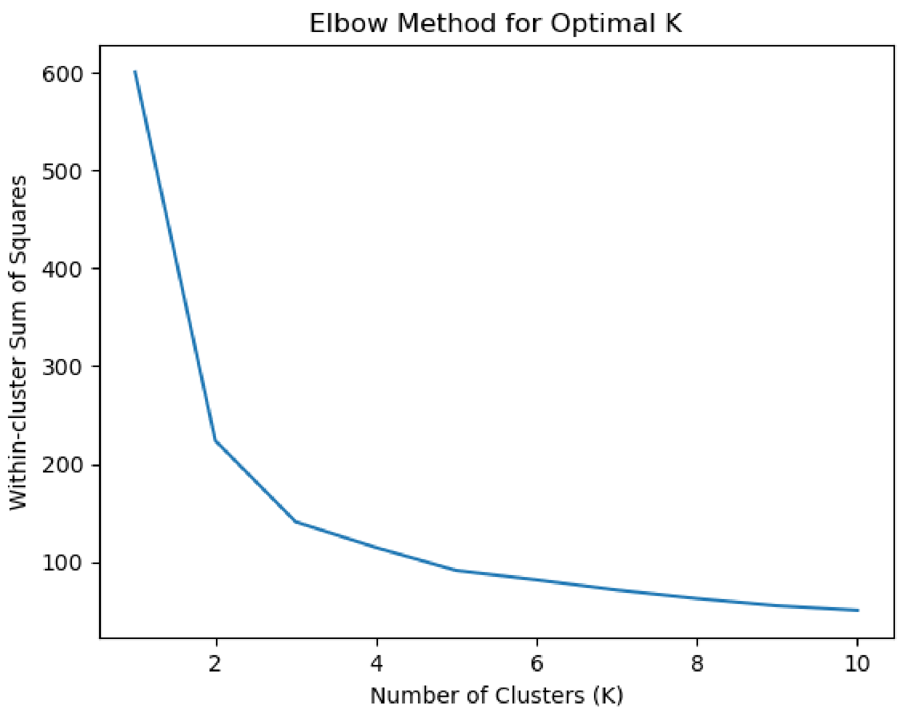
In [45]:
```
pca = PCA(n_components=2)
features_pca = pca.fit_transform(features_scaled)
print(f"features_pca.shape {features_pca.shape}, first few entries \n{features_pca[:5]}")
```

```
features_pca.shape (150, 2), first few entries
[[-2.26454173  0.5057039 ]
 [-2.0864255  -0.65540473]
 [-2.36795045 -0.31847731]
 [-2.30419716 -0.57536771]
 [-2.38877749  0.6747674 ]]
```

In [46]:
```
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, random_state=42)
    kmeans.fit(features_scaled)
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss)
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Within-cluster Sum of Squares')
plt.show()
```

```
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

uto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(



k means Algorithm

Split the Data

In [47]:
```
X_train, X_test, y_train, y_test = train_test_split(features_pca, target, test_size=0.2, random_state=42)
```

In [48]:
```
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_train)
```

C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\A.Surya Tejaswini\.conda\Lib\site-packages\sklearn\cluster_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(

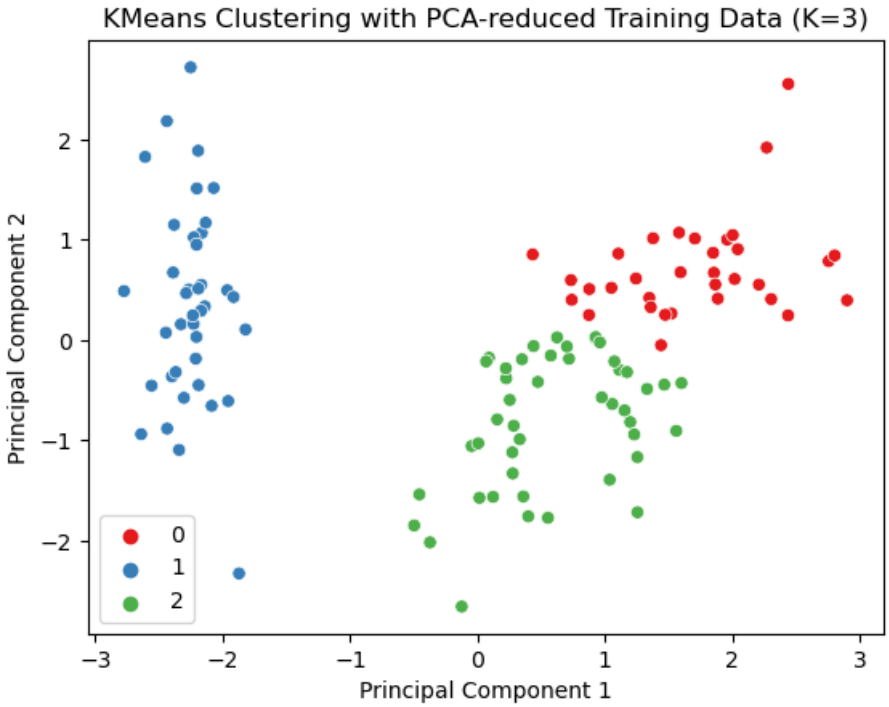
Out[48]:

▾ KMeans

KMeans(n_clusters=3, random_state=42)

In [49]:
```
sns.scatterplot(x=X_train[:, 0], y=X_train[:, 1], hue=kmeans.labels_, palette='Set1')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
```

```
plt.title('KMeans Clustering with PCA-reduced Training Data (K=3)')
plt.show()
```



In [50]:
```
train_y_predict = kmeans.predict(X_train)
```

In [51]:
```
test_y_predict = kmeans.predict(X_test)
```

In [52]:
```
train_score = silhouette_score(X_train, train_y_predict)
test_score = silhouette_score(X_test, test_y_predict)

print("Training Score:", train_score)
print("Testing Score:", test_score)
```

```
Training Score: 0.5170233047741806
Testing Score: 0.47809419718070867
```