The Iris dataset is one of the most famous datasets in the field of machine learning and statistics. It was introduced by the British biologist and statistician Ronald A. Fisher in 1936. The dataset contains 150 observations of iris flowers, with three different species (Iris setosa, Iris versicolor, and Iris virginica). Each observation includes four features measured from each flower: sepal length, sepal width, petal length, and petal width. Here is a summary of the Iris dataset:

Features:

1. **Sepal Length** (in centimeters)
2. **Sepal Width** (in centimeters)
3. **Petal Length** (in centimeters)
4. **Petal Width** (in centimeters)

Target Variable:

- **Species**:
  - Iris setosa
  - Iris versicolor
  - Iris virginica

Dataset Composition:

- **Total Observations**: 150
- **Observations per Species**: 50

Characteristics:

- **Class Distribution**: The dataset is balanced with each class (species) having 50 instances.
- **Feature Relationships**: The features are continuous and can be used to distinguish between the three species of iris flowers. For example, Iris setosa is easily separable from the other two species based on petal length and petal width.

Visualization:

- **Pair Plot**: A pair plot (scatter plot matrix) is commonly used to visualize the relationships between different features and the separability of the classes.
- **Box Plot**: Box plots can be used to show the distribution of each feature for the different species.

Usage:

The Iris dataset is commonly used for:

- Classification tasks
- Demonstrating algorithms in machine learning courses
- Evaluating and comparing the performance of machine learning models

Summary:

The Iris dataset is a classic example of a simple, well-balanced dataset with clearly distinguishable classes. It provides an excellent introduction to classification problems and is often used to illustrate the application of various machine learning algorithms.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pylab as plt
%matplotlib inline

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

from scipy.stats import norm
from scipy import stats

data=pd.read_csv("/kaggle/input/irisdata/Iris.csv")
data.head(10)
```

Out[1]:

	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
5	6	5.4	3.9	1.7	0.4	Iris-setosa

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

In [2]:

```
print(data.shape[0])
```

```
150
```

In [3]:

```
print(data.columns.tolist())
```

```
['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm', 'Spec
ies']
```

In [4]:

```
print(data.dtypes)
```

```
Id                 int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
```

In [5]:

```
#drop Id column
df = data.drop('Id', axis=1)
print(df)
```

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm         Species
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 x 5 columns]
```

In [6]:

```python
# Select just the rows desired from the 'describe' method and add in the 'median'
df.describe()
```

Out[6]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
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 [7]:

```python
# The str method maps the following function to each entry as a string
df['Species'] =df.Species.str.replace('Iris-', '')
```

In [8]:

```python
df.head(10)
```

Out[8]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	setosa

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

In [9]:

```
#check for the count of each species
df.Species.value_counts()
```

Out[9]:

```
Species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

In [10]:

```
stats_df=df.describe()

stats_df.loc['range']=stats_df.loc['max']-stats_df.loc['min']
print(stats_df)
```

```
       SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
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
range      3.600000     2.400000      5.900000      2.400000
```

In [11]:

```
#check for null values
null_values = df.isnull()
print(null_values)
```

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  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 x 5 columns]
```

In [12]:

```
df_Mean=df.groupby('Species').mean()
df_Mean_median=df.groupby('Species').agg([np.mean, np.median])
df_Mean_median
```

Out[12]:

	SepalLengthCm		SepalWidthCm		PetalLengthCm		PetalWidthCm	
	mean	median	mean	median	mean	median	mean	median
Species								
setosa	5.006	5.0	3.418	3.4	1.464	1.50	0.244	0.2
versicolor	5.936	5.9	2.770	2.8	4.260	4.35	1.326	1.3
virginica	6.588	6.5	2.974	3.0	5.552	5.55	2.026	2.0

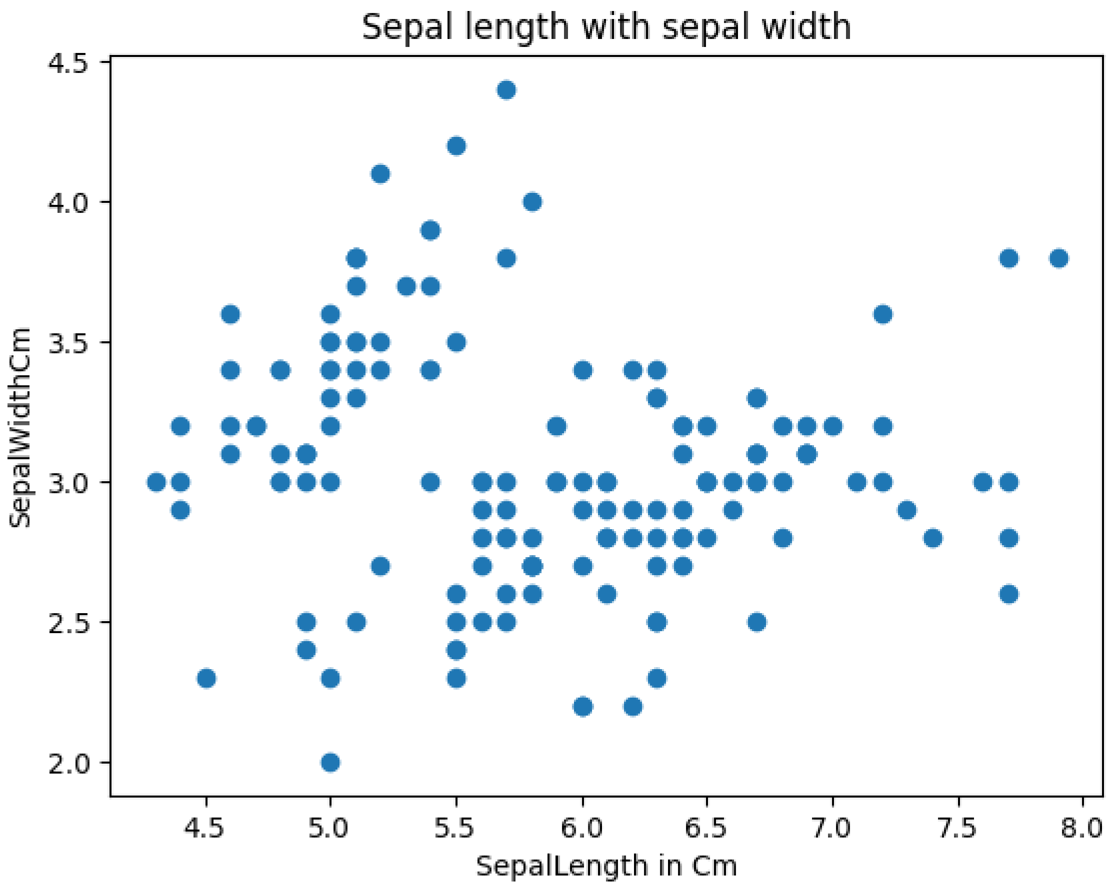
In [13]:

```
#scatter plot for sepal length and sepal width to check for linear relation
```

```
ax=plt.axes()
ax.scatter(df.SepalLengthCm,df.SepalWidthCm)
ax.set(xlabel='SepalLength in Cm',ylabel='SepalWidthCm',title='Sepal length with s
epal width')
```

Out[13]:

```
[Text(0.5, 0, 'SepalLength in Cm'),
 Text(0, 0.5, 'SepalWidthCm'),
 Text(0.5, 1.0, 'Sepal length with sepal width')]
```

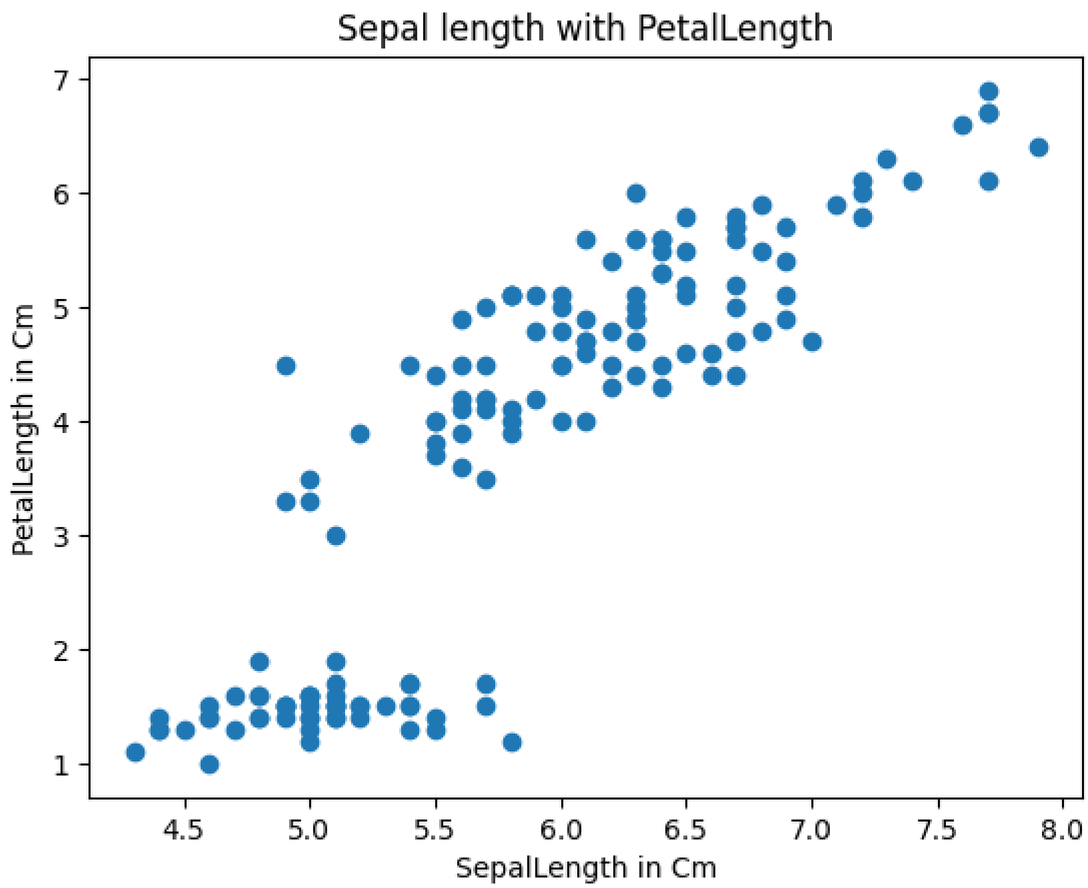


In [14]:

```
#scatter plot for sepal length and sepal width to check for linear relation
ax=plt.axes()
ax.scatter(df.SepalLengthCm,df.PetalLengthCm)
ax.set(xlabel='SepalLength in Cm',ylabel='PetalLength in Cm',title='Sepal length w
ith PetalLength')
```

Out[14]:

```
[Text(0.5, 0, 'SepalLength in Cm'),
 Text(0, 0.5, 'PetalLength in Cm'),
 Text(0.5, 1.0, 'Sepal length with PetalLength')]
```

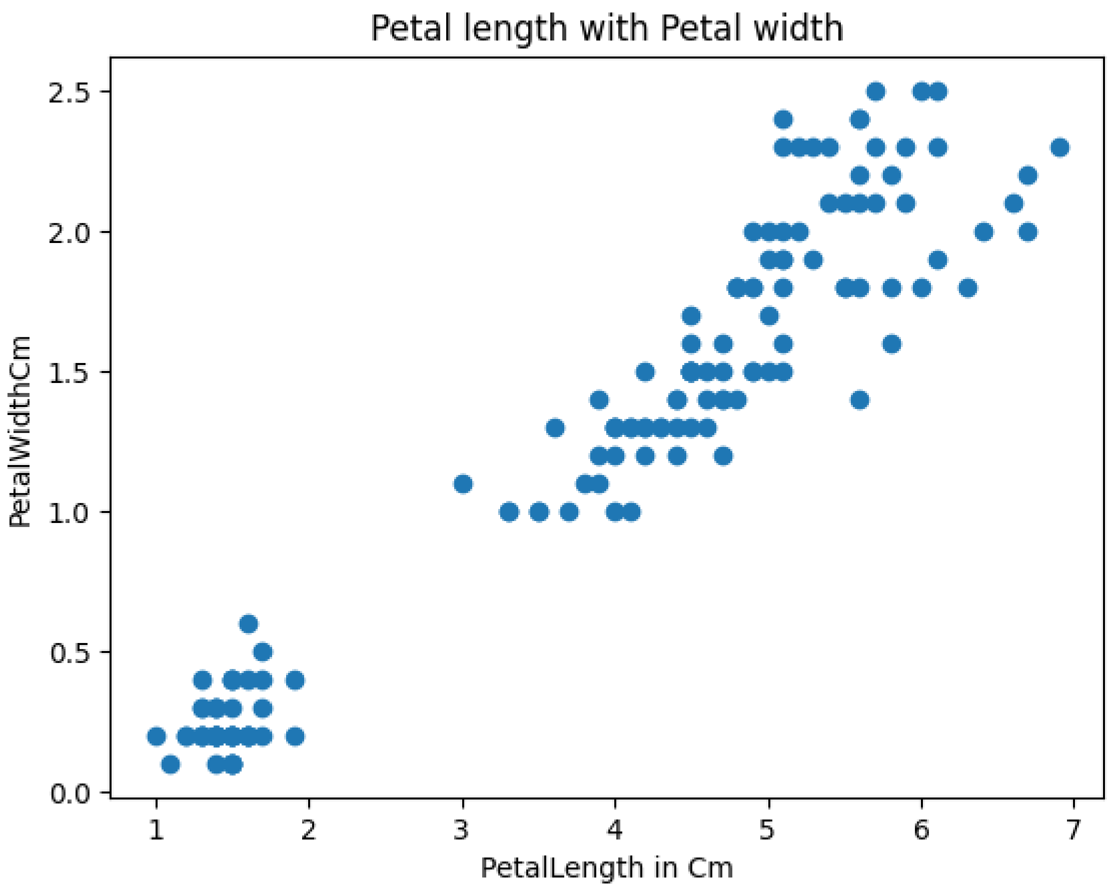


In [15]:

```
#scatter plot for petal length and petal width to check for linear relation
ax=plt.axes()
ax.scatter(df.PetalLengthCm,df.PetalWidthCm)
ax.set(xlabel='PetalLength in Cm',ylabel='PetalWidthCm',title='Petal length with P
etal width')
```

Out[15]:

```
[Text(0.5, 0, 'PetalLength in Cm'),
 Text(0, 0.5, 'PetalWidthCm'),
 Text(0.5, 1.0, 'Petal length with Petal width')]
```

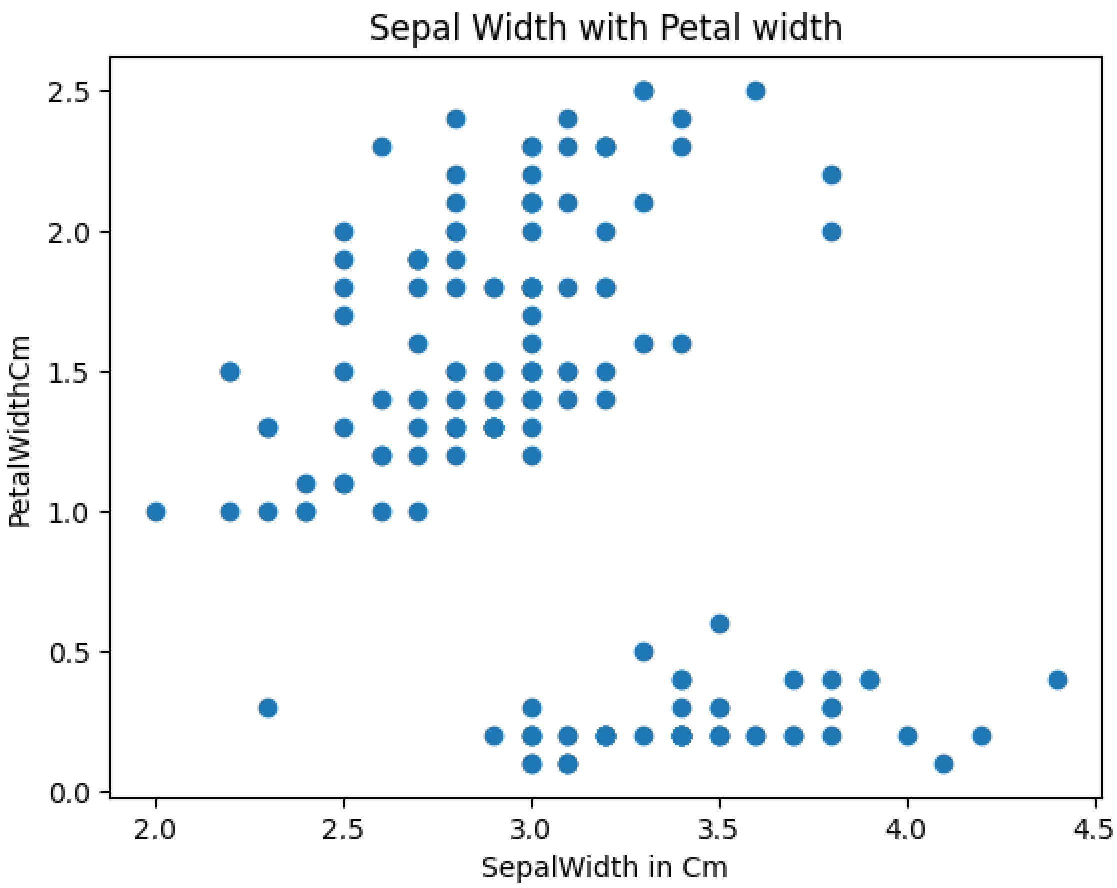


In [16]:

```
#scatter plot for petal length and petal width to check for linear relation
ax=plt.axes()
ax.scatter(df.SepalWidthCm,df.PetalWidthCm)
ax.set(xlabel='SepalWidth in Cm',ylabel='PetalWidthCm',title='Sepal Width with Pet
al width')
```

Out[16]:

```
[Text(0.5, 0, 'SepalWidth in Cm'),
 Text(0, 0.5, 'PetalWidthCm'),
 Text(0.5, 1.0, 'Sepal Width with Petal width')]
```



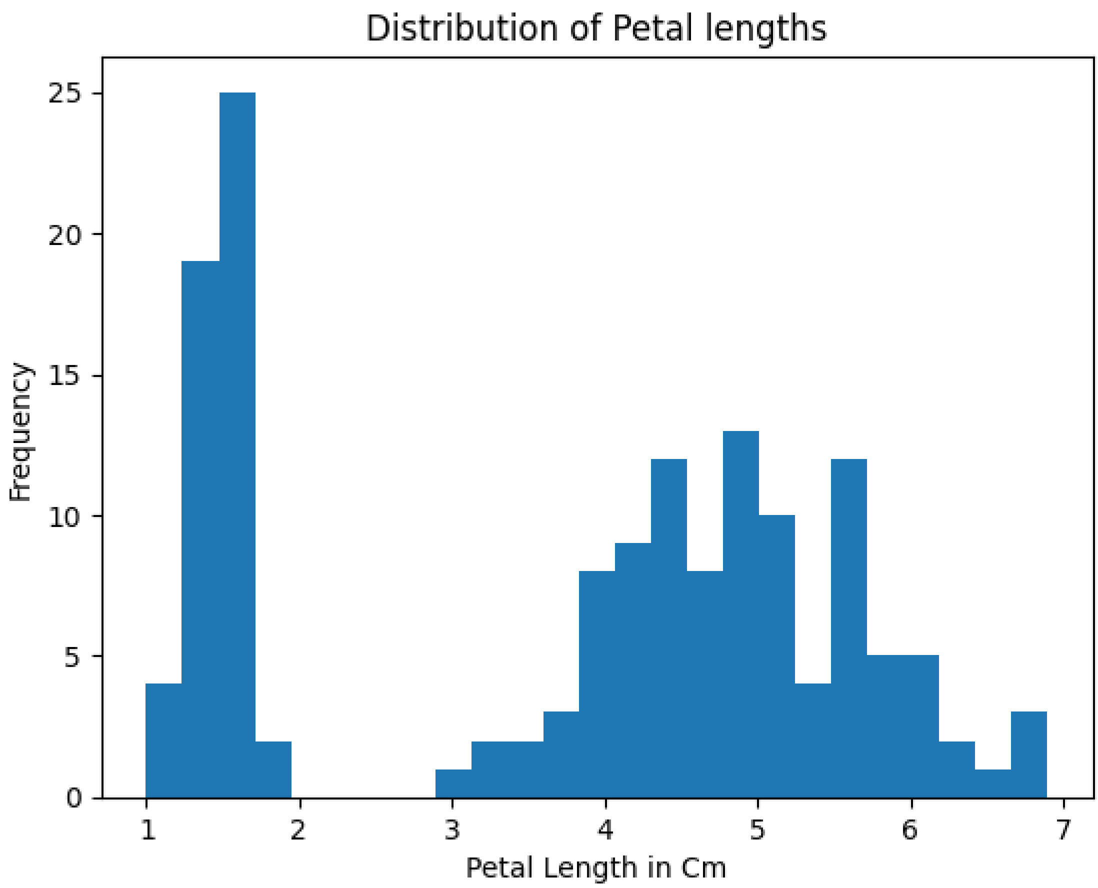
In [17]:

```
#there is linear relation between petal length and petal width
#there is linear relation between petal length and sepal length

#plot a histogram to check for normality distribution for all
ax=plt.axes()
ax.hist(df.PetalLengthCm, bins=25)
ax.set(xlabel='Petal Length in Cm',ylabel='Frequency', title='Distribution of Peta
l lengths')
```

Out[17]:

```
[Text(0.5, 0, 'Petal Length in Cm'),
 Text(0, 0.5, 'Frequency'),
 Text(0.5, 1.0, 'Distribution of Petal lengths')]
```

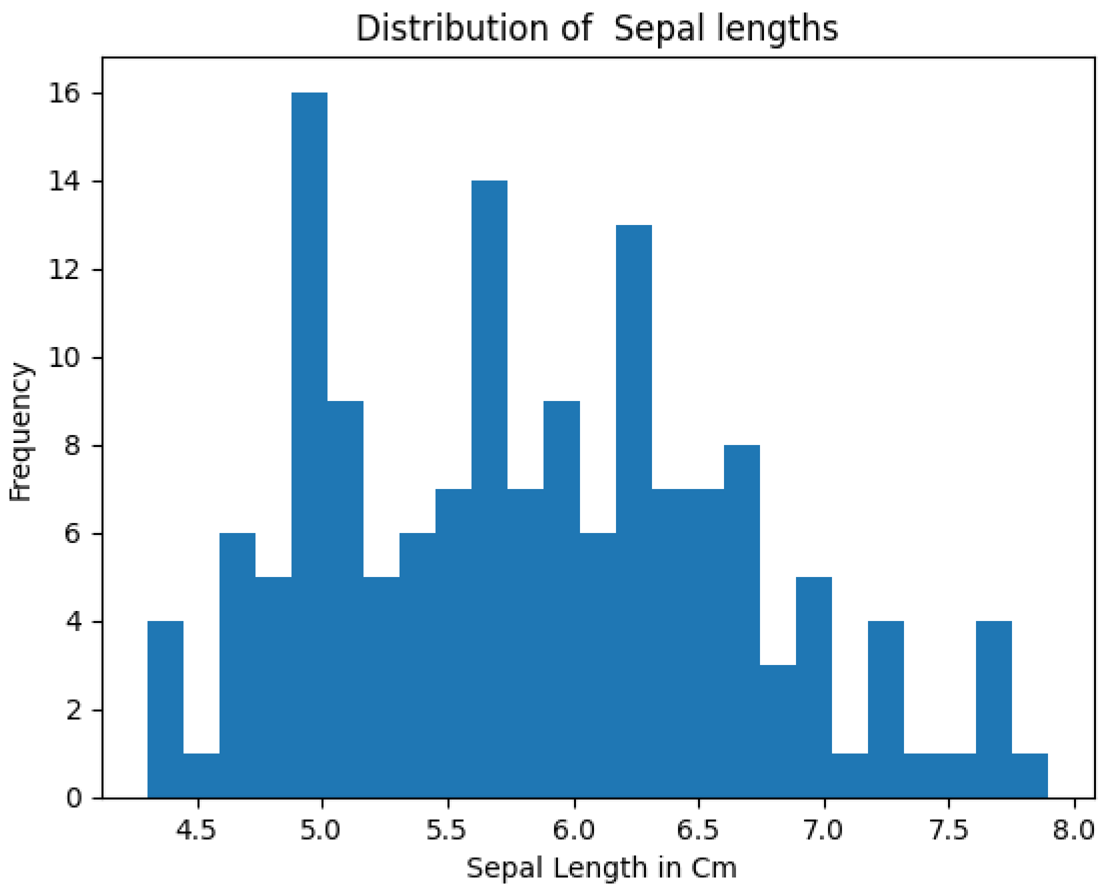


In [18]:

```
ax=plt.axes()
ax.hist(df.SepalLengthCm, bins=25)
ax.set(xlabel=' Sepal Length in Cm',ylabel='Frequency', title='Distribution of  Se
pal lengths')
```

Out[18]:

```
[Text(0.5, 0, ' Sepal Length in Cm'),
 Text(0, 0.5, 'Frequency'),
 Text(0.5, 1.0, 'Distribution of  Sepal lengths')]
```

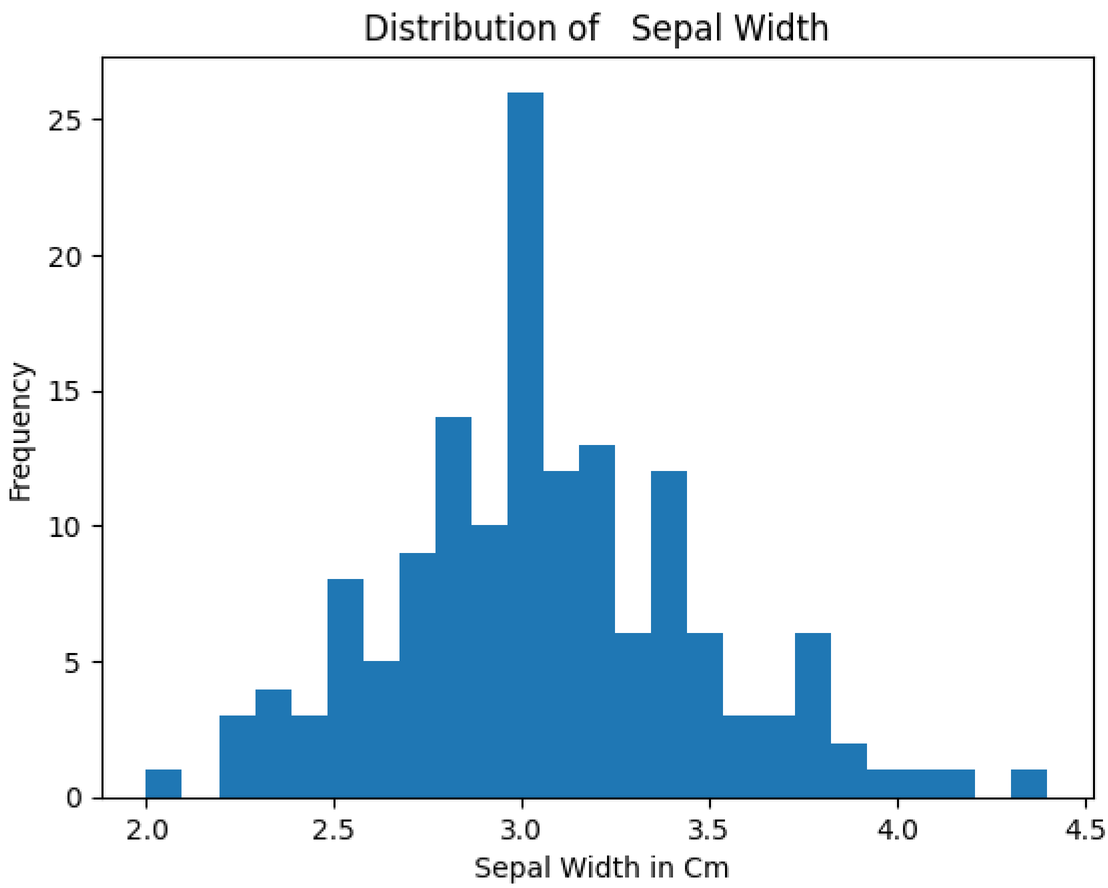


In [19]:

```
ax=plt.axes()
ax.hist(df.SepalWidthCm, bins=25)
ax.set(xlabel=' Sepal Width in Cm',ylabel='Frequency', title='Distribution of   Se
pal Width')
```

Out[19]:

```
[Text(0.5, 0, ' Sepal Width in Cm'),
 Text(0, 0.5, 'Frequency'),
 Text(0.5, 1.0, 'Distribution of   Sepal Width')]
```

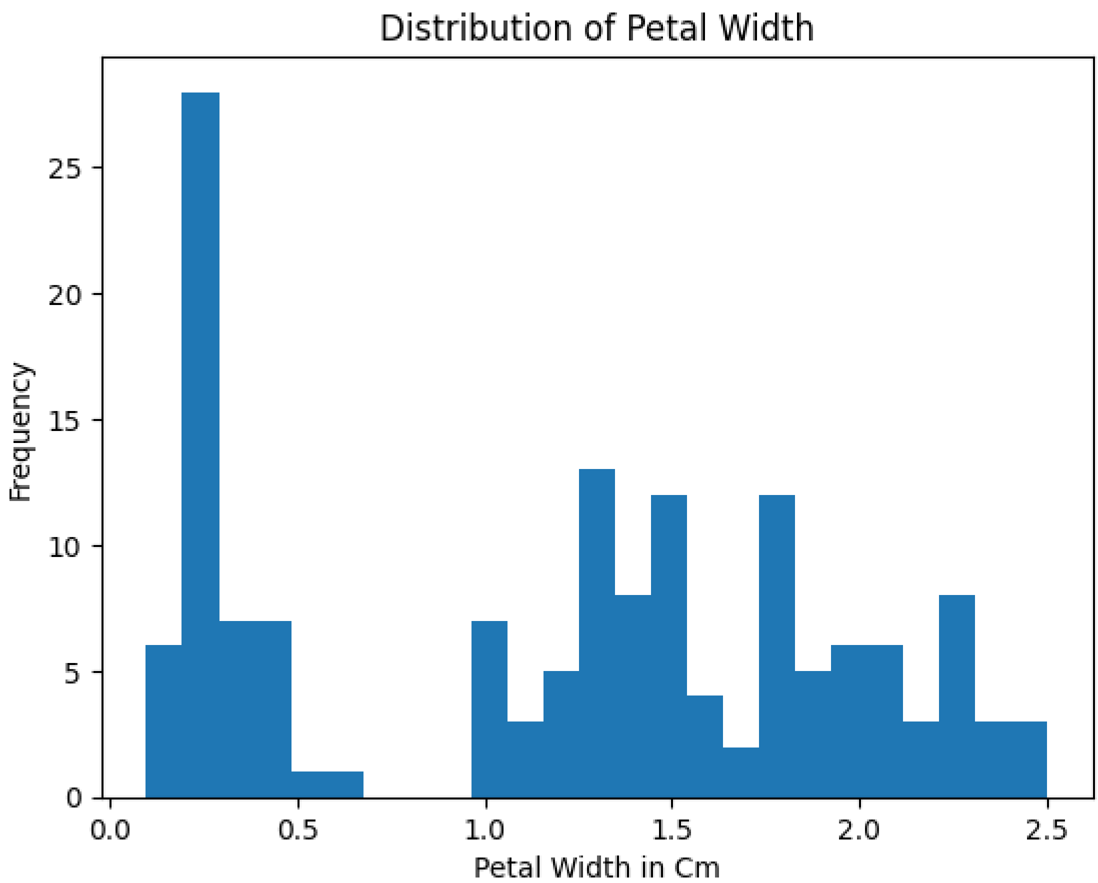


In [20]:

```
ax=plt.axes()
ax.hist(df.PetalWidthCm, bins=25)
ax.set(xlabel='Petal Width in Cm',ylabel='Frequency', title='Distribution of Petal 
Width')
```

Out[20]:

```
[Text(0.5, 0, 'Petal Width in Cm'),
 Text(0, 0.5, 'Frequency'),
 Text(0.5, 1.0, 'Distribution of Petal Width')]
```

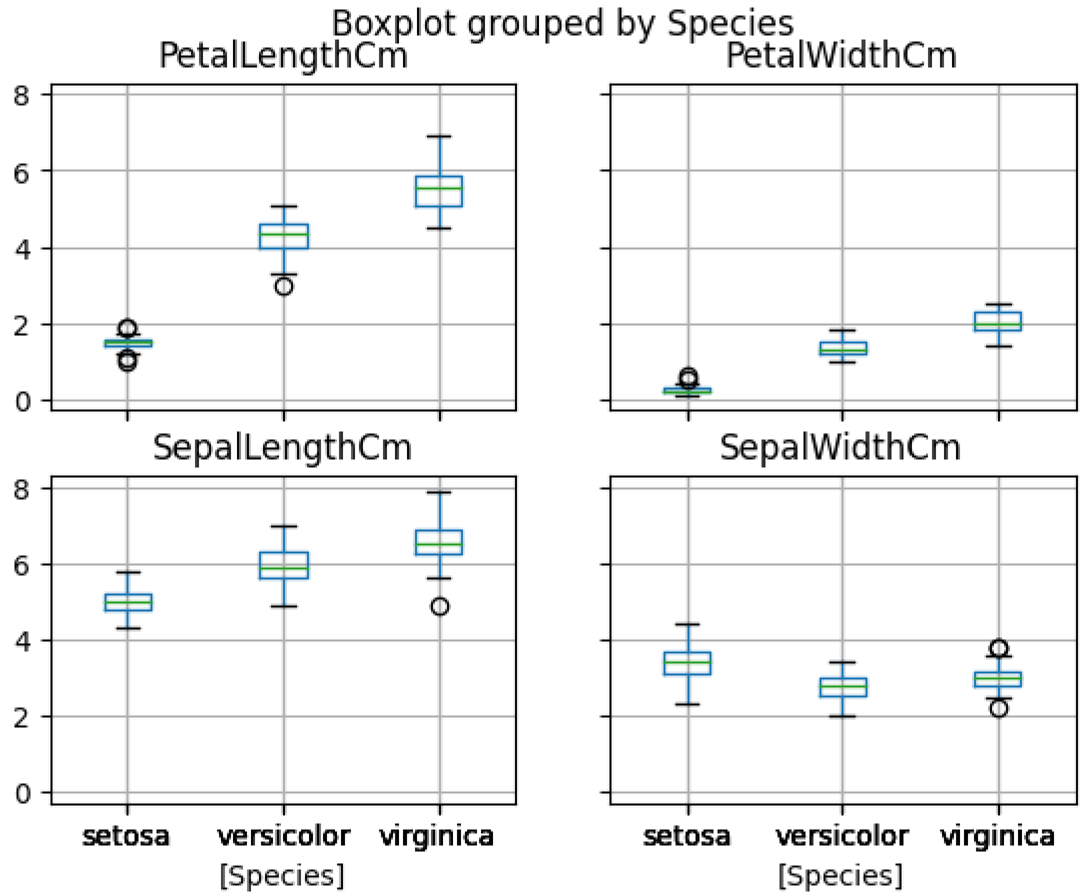


In [21]:

```
#All follow a normal distribution
#create box plot for each species
df.boxplot(by='Species')
```

Out[21]:

```
array([[<Axes: title={'center': 'PetalLengthCm'}, xlabel='[Species]'>,
        <Axes: title={'center': 'PetalWidthCm'}, xlabel='[Species]'>],
       [<Axes: title={'center': 'SepalLengthCm'}, xlabel='[Species]'>,
        <Axes: title={'center': 'SepalWidthCm'}, xlabel='[Species]'>]],
      dtype=object)
```



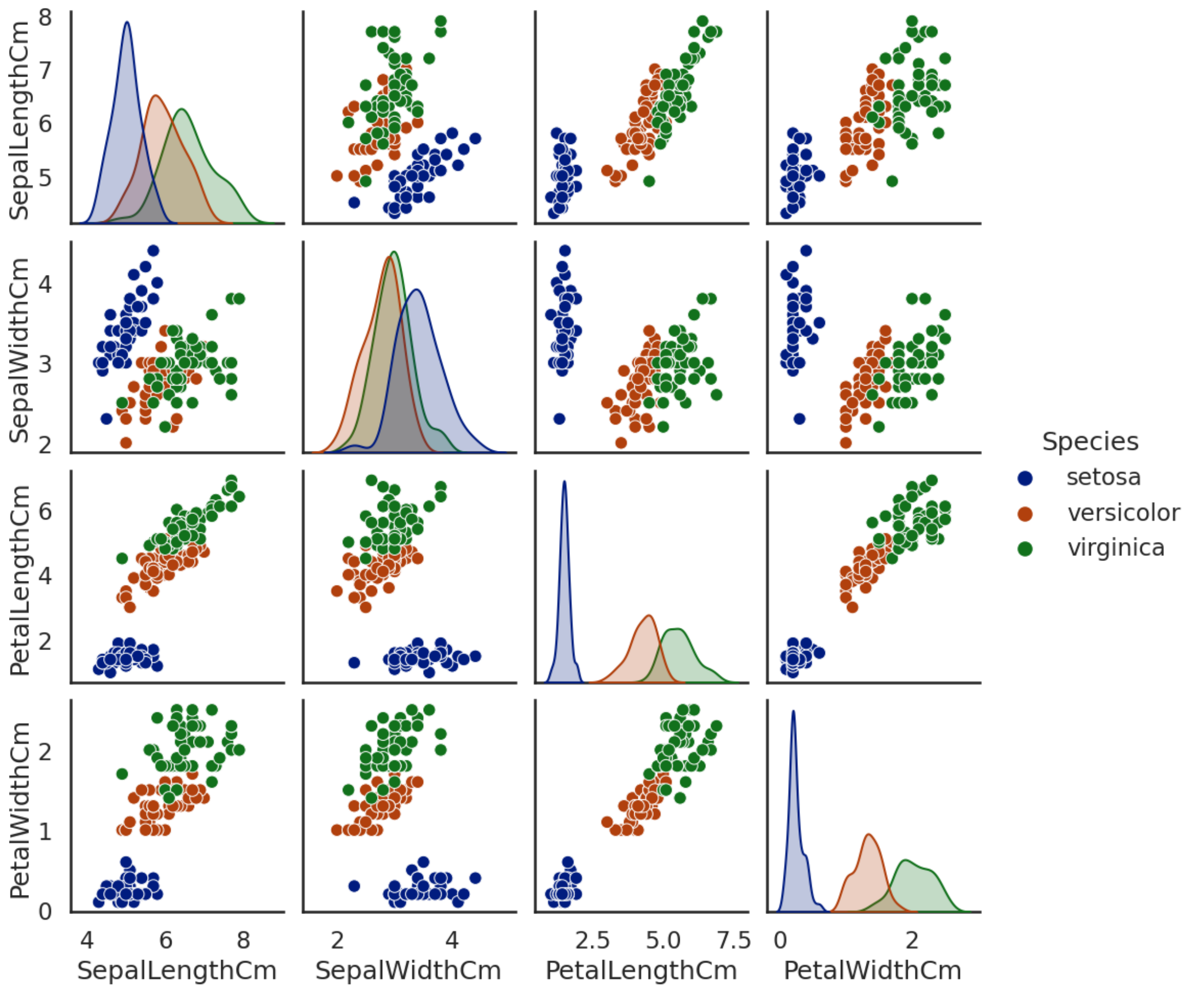
In [22]:

```
#PAIR PLOT

sns.set_style('white')
sns.set_context('talk')
sns.set_palette('dark')
sns.pairplot(data=df, hue='Species')
```

Out[22]:

```
<seaborn.axisgrid.PairGrid at 0x7b7c3c60bfa0>
```



In [23]:

```
#For Species Setosa all the features are not correlated but sepal length and width
are correlated.
#For Versiclor species petal width and sepal width are correlated, petal width and
petal length are linearly correlated.
#Petal length is linearly correlated with sepal length, petal length and sepal wid
th are correlated. Sepal length and sepal width are correlated.
#For virginica species species petal width and sepal width are correlated, petal w
idth and petal length are linearly correlated.
#Petal length is linearly correlated with sepal length, petal length and sepal wid
th are correlated. Sepal length and sepal width are correlated.
# For each of the Species. All the features( petal width ,sepal width, Petal lengt
h and sepal length)follow normal distribution.
```

In [24]:

```
df.head(10)
```

Out[24]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	setosa

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
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
5	5.4	3.9	1.7	0.4	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa

In [25]:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
#lets separate data from target column

X = df.drop(columns = 'Species') # alternatively we can use >> df.drop('Species' ,
axis = 1)
Y = df['Species']

X.shape
```

Out[25]:

```
(150, 4)
```

In [26]:

```
Y.shape
```

Out[26]:

```
(150,)
```

In [27]:

```
arr = scaler.fit_transform(X)
X_scaled = pd.DataFrame(arr, columns = X.columns)
X_scaled
```

Out[27]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	0.222222	0.625000	0.067797	0.041667
1	0.166667	0.416667	0.067797	0.041667
2	0.111111	0.500000	0.050847	0.041667
3	0.083333	0.458333	0.084746	0.041667
4	0.194444	0.666667	0.067797	0.041667
...	...	...	...	...
145	0.666667	0.416667	0.711864	0.916667
146	0.555556	0.208333	0.677966	0.750000
147	0.611111	0.416667	0.711864	0.791667
148	0.527778	0.583333	0.745763	0.916667
149	0.444444	0.416667	0.694915	0.708333

150 rows × 4 columns

In [28]:

```
from sklearn.model_selection import train_test_split
```

```python
from sklearn.model_selection import GridSearchCV

x_train, x_test, y_train, y_test = train_test_split(X_scaled, Y, train_size = 0.75
, random_state = 5)

x_train.shape , x_test.shape , y_train.shape , y_test.shape
```

Out[28]:

```
((112, 4), (38, 4), (112,), (38,))
```

In [29]:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report , accuracy_score
from sklearn.metrics import multilabel_confusion_matrix
```

In [30]:

```python
log_model = LogisticRegression(multi_class = 'ovr')
log_model.fit(x_train , y_train)
```

```
LogisticRegression(multi_class='ovr')
```

Out[30]:

```
LogisticRegression
LogisticRegression(multi_class='ovr')
```

In [31]:

```python
x_train
```

Out[31]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
40	0.194444	0.625000	0.050847	0.083333
115	0.583333	0.500000	0.728814	0.916667
142	0.416667	0.291667	0.694915	0.750000
69	0.361111	0.208333	0.491525	0.416667
17	0.222222	0.625000	0.067797	0.083333
...	...	...	...	...

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
8	0.027778	0.375000	0.067797	0.041667
73	0.500000	0.333333	0.627119	0.458333
144	0.666667	0.541667	0.796610	1.000000
118	0.944444	0.250000	1.000000	0.916667
99	0.388889	0.333333	0.525424	0.500000

112 rows × 4 columns

In [32]:

```
x_test
```

Out[32]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
82	0.416667	0.291667	0.491525	0.458333
134	0.500000	0.250000	0.779661	0.541667
114	0.416667	0.333333	0.694915	0.958333
42	0.027778	0.500000	0.050847	0.041667
109	0.805556	0.666667	0.864407	1.000000
57	0.166667	0.166667	0.389831	0.375000

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
1	0.166667	0.416667	0.067797	0.041667
70	0.444444	0.500000	0.644068	0.708333
25	0.194444	0.416667	0.101695	0.041667
84	0.305556	0.416667	0.593220	0.583333
66	0.361111	0.416667	0.593220	0.583333
133	0.555556	0.333333	0.694915	0.583333
102	0.777778	0.416667	0.830508	0.833333
107	0.833333	0.375000	0.898305	0.708333
26	0.194444	0.583333	0.101695	0.125000
23	0.222222	0.541667	0.118644	0.166667
123	0.555556	0.291667	0.661017	0.708333
130	0.861111	0.333333	0.864407	0.750000
21	0.222222	0.708333	0.084746	0.125000

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
12	0.138889	0.416667	0.067797	0.000000
71	0.500000	0.333333	0.508475	0.500000
128	0.583333	0.333333	0.779661	0.833333
48	0.277778	0.708333	0.084746	0.041667
72	0.555556	0.208333	0.661017	0.583333
88	0.361111	0.416667	0.525424	0.500000
148	0.527778	0.583333	0.745763	0.916667
74	0.583333	0.375000	0.559322	0.500000
96	0.388889	0.375000	0.542373	0.500000
63	0.500000	0.375000	0.627119	0.541667
132	0.583333	0.333333	0.779661	0.875000
39	0.222222	0.583333	0.084746	0.041667
53	0.333333	0.125000	0.508475	0.500000

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
79	0.388889	0.250000	0.423729	0.375000
10	0.305556	0.708333	0.084746	0.041667
50	0.750000	0.500000	0.627119	0.541667
49	0.194444	0.541667	0.067797	0.041667
43	0.194444	0.625000	0.101695	0.208333
135	0.944444	0.416667	0.864407	0.916667

In [33]:

```
y_train
```

Out[33]:

```
40         setosa
115     virginica
142     virginica
69     versicolor
17         setosa
          ...
8          setosa
73     versicolor
144     virginica
118     virginica
99     versicolor
Name: Species, Length: 112, dtype: object
```

In [34]:

```
y_test
```

Out[34]:

```
82     versicolor
134     virginica
114     virginica
42         setosa
109     virginica
57     versicolor
1          setosa
70     versicolor
25         setosa
84     versicolor
```

```
66     versicolor
133     virginica
102     virginica
107     virginica
26         setosa
23         setosa
123     virginica
130     virginica
21         setosa
12         setosa
71     versicolor
128     virginica
48         setosa
72     versicolor
88     versicolor
148     virginica
74     versicolor
96     versicolor
63     versicolor
132     virginica
39         setosa
53     versicolor
79     versicolor
10         setosa
50     versicolor
49         setosa
43         setosa
135     virginica
Name: Species, dtype: object
```

In [35]:

```
# Predicting labels on the training dataset
y_pred_train = log_model.predict(x_train)
```

In [36]:

```
# Calculating accuracy
accuracy = round(accuracy_score(y_train, y_pred_train), 2)

# Generating a multilabel confusion matrix
conf_mat = multilabel_confusion_matrix(y_train, y_pred_train)

# Generating a classification report
class_rep = classification_report(y_train, y_pred_train)

# Printing the results

print(f'Accuracy of the model on the training data is: {accuracy}')

print(f'Classification Report of the model on training data:\n{class_rep}')

print(f'Multilabel Confusion Matrix:\n\n{conf_mat}')
```

```
Accuracy of the model on the training data is: 0.86
Classification Report of the model on training data:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        38
  versicolor       0.92      0.61      0.73        36
   virginica       0.72      0.95      0.82        38
```

```
    accuracy                           0.86       112
   macro avg       0.88      0.85      0.85       112
weighted avg       0.88      0.86      0.85       112

Multilabel Confusion Matrix:

[[[74  0]
  [ 0 38]]

 [[74  2]
  [14 22]]

 [[60 14]
  [ 2 36]]]
```

In [37]:

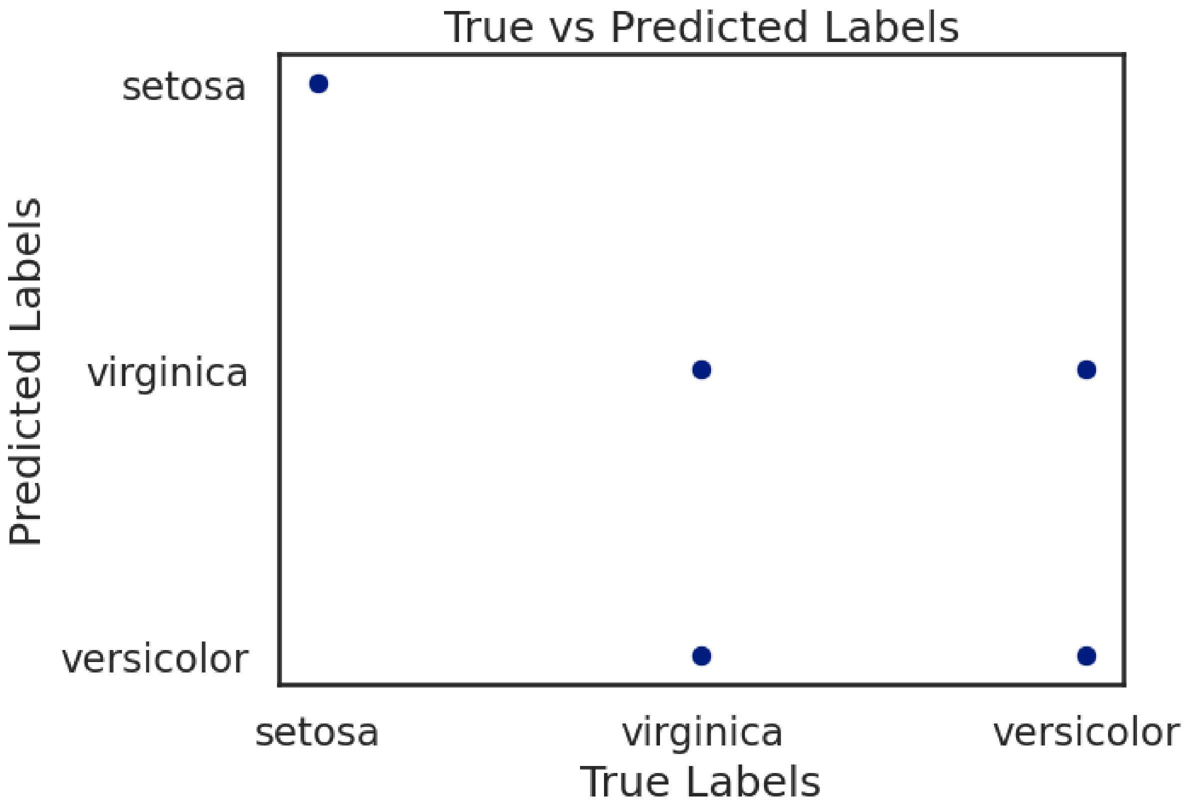
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming y_train and y_pred_train are Pandas Series or NumPy arrays
df_plot = pd.DataFrame({'True Labels': y_train, 'Predicted Labels': y_pred_train})

# Create a scatter plot
sns.scatterplot(x='True Labels', y='Predicted Labels', data=df_plot)

# Add labels and title
plt.xlabel('True Labels')
plt.ylabel('Predicted Labels')
plt.title('True vs Predicted Labels')

# Show the plot
plt.show()
```



In [38]:

```
#Accuracy of LogisticRegression Model on Testing Dataset

y_pred = log_model.predict(x_test)

accuracy = round(accuracy_score(y_test , y_pred), 2)

conf_mat = multilabel_confusion_matrix(y_test , y_pred)

class_rep = classification_report(y_test , y_pred)
print(f'Accuracy of model on testing data is: {accuracy}')
print(f'Classification Report of the model on training data:\n{class_rep}')
print(f'Multilabel Confusion Matrix : \n\n {conf_mat}')
```

```
Accuracy of model on testing data is: 0.84
Classification Report of the model on training data:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        12
  versicolor       0.90      0.64      0.75        14
   virginica       0.69      0.92      0.79        12

    accuracy                           0.84        38
   macro avg       0.86      0.85      0.85        38
weighted avg       0.86      0.84      0.84        38

Multilabel Confusion Matrix :

 [[[26  0]
  [ 0 12]]

 [[23  1]
```

```
  [ 5  9]]

 [[21  5]
  [ 1 11]]]
```

In [39]:

```
y_test
```

Out[39]:

```
82     versicolor
134     virginica
114     virginica
42         setosa
109     virginica
57     versicolor
1          setosa
70     versicolor
25         setosa
84     versicolor
66     versicolor
133     virginica
102     virginica
107     virginica
26         setosa
23         setosa
123     virginica
130     virginica
21         setosa
12         setosa
71     versicolor
128     virginica
48         setosa
72     versicolor
88     versicolor
148     virginica
74     versicolor
96     versicolor
63     versicolor
132     virginica
39         setosa
53     versicolor
79     versicolor
10         setosa
50     versicolor
49         setosa
43         setosa
135     virginica
Name: Species, dtype: object
```

In [40]:

```
y_pred
```

Out[40]:

```
array(['versicolor', 'versicolor', 'virginica', 'setosa', 'virginica',
       'versicolor', 'setosa', 'virginica', 'setosa', 'virginica',
       'virginica', 'virginica', 'virginica', 'virginica', 'setosa',
       'setosa', 'virginica', 'virginica', 'setosa', 'setosa',
       'versicolor', 'virginica', 'setosa', 'versicolor', 'versicolor',
       'virginica', 'versicolor', 'versicolor', 'virginica', 'virginica',
```

```
       'setosa', 'versicolor', 'versicolor', 'setosa', 'virginica',
       'setosa', 'setosa', 'virginica'], dtype=object)
```

In [41]:

```
x_test
```

Out[41]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
82	0.416667	0.291667	0.491525	0.458333
134	0.500000	0.250000	0.779661	0.541667
114	0.416667	0.333333	0.694915	0.958333
42	0.027778	0.500000	0.050847	0.041667
109	0.805556	0.666667	0.864407	1.000000
57	0.166667	0.166667	0.389831	0.375000
1	0.166667	0.416667	0.067797	0.041667
70	0.444444	0.500000	0.644068	0.708333
25	0.194444	0.416667	0.101695	0.041667
84	0.305556	0.416667	0.593220	0.583333
66	0.361111	0.416667	0.593220	0.583333
133	0.555556	0.333333	0.694915	0.583333

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
102	0.777778	0.416667	0.830508	0.833333
107	0.833333	0.375000	0.898305	0.708333
26	0.194444	0.583333	0.101695	0.125000
23	0.222222	0.541667	0.118644	0.166667
123	0.555556	0.291667	0.661017	0.708333
130	0.861111	0.333333	0.864407	0.750000
21	0.222222	0.708333	0.084746	0.125000
12	0.138889	0.416667	0.067797	0.000000
71	0.500000	0.333333	0.508475	0.500000
128	0.583333	0.333333	0.779661	0.833333
48	0.277778	0.708333	0.084746	0.041667
72	0.555556	0.208333	0.661017	0.583333
88	0.361111	0.416667	0.525424	0.500000

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
148	0.527778	0.583333	0.745763	0.916667
74	0.583333	0.375000	0.559322	0.500000
96	0.388889	0.375000	0.542373	0.500000
63	0.500000	0.375000	0.627119	0.541667
132	0.583333	0.333333	0.779661	0.875000
39	0.222222	0.583333	0.084746	0.041667
53	0.333333	0.125000	0.508475	0.500000
79	0.388889	0.250000	0.423729	0.375000
10	0.305556	0.708333	0.084746	0.041667
50	0.750000	0.500000	0.627119	0.541667
49	0.194444	0.541667	0.067797	0.041667
43	0.194444	0.625000	0.101695	0.208333
135	0.944444	0.416667	0.864407	0.916667

In [42]:

```
import seaborn as sns
```

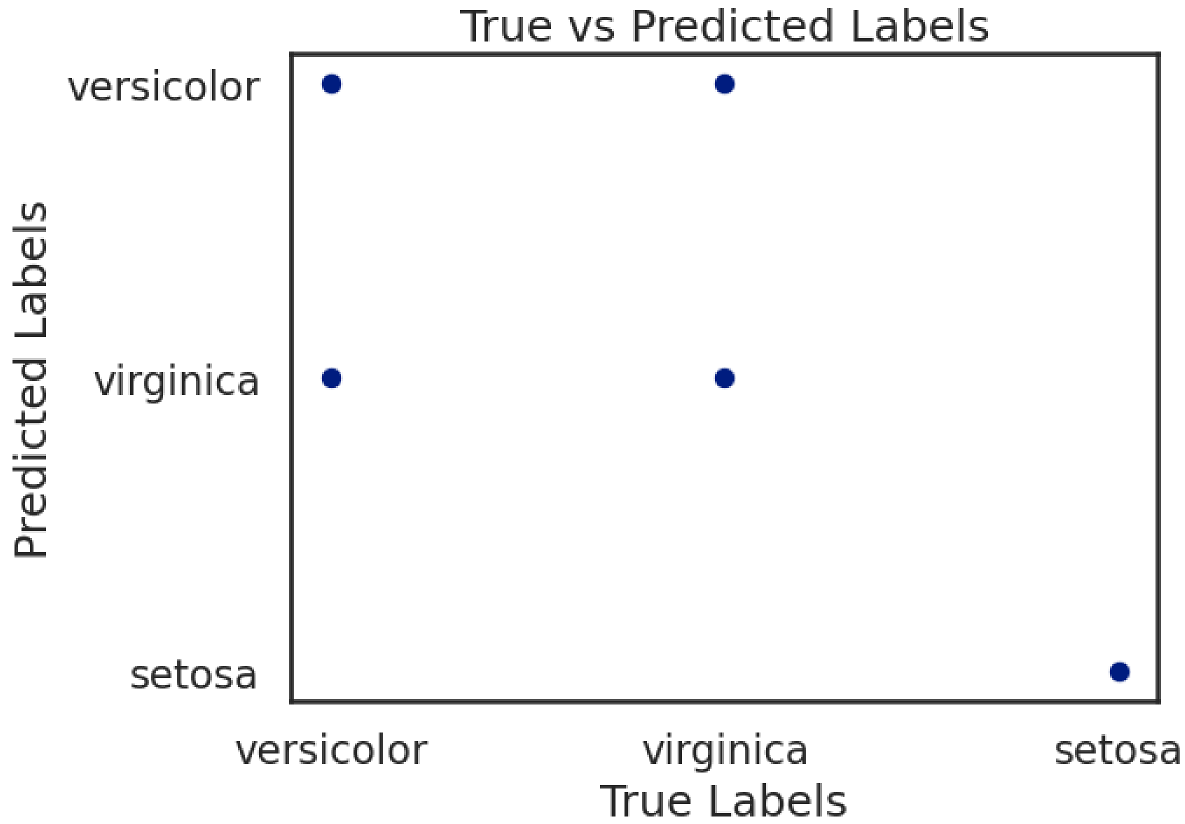
```python
import matplotlib.pyplot as plt

# Assuming y_train and y_pred_train are Pandas Series or NumPy arrays
df_plot = pd.DataFrame({'True Labels': y_test, 'Predicted Labels': y_pred})

# Create a scatter plot
sns.scatterplot(x='True Labels', y='Predicted Labels', data=df_plot)

# Add labels and title
plt.xlabel('True Labels')
plt.ylabel('Predicted Labels')
plt.title('True vs Predicted Labels')

# Show the plot
plt.show()
```



In [ ]: