In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

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

In [3]:
```
df
```

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
...	...	...	...	...	...	...
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 [4]:
```
column = df.columns
for columns in column:
    print(columns)
```

```
Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm
Species
```

In [5]:
```
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
```

In [6]:
```
df.describe()
```

Out[6]:

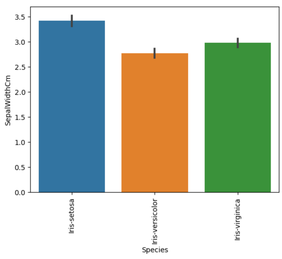
	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	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.isnull().sum()
```

Out[7]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

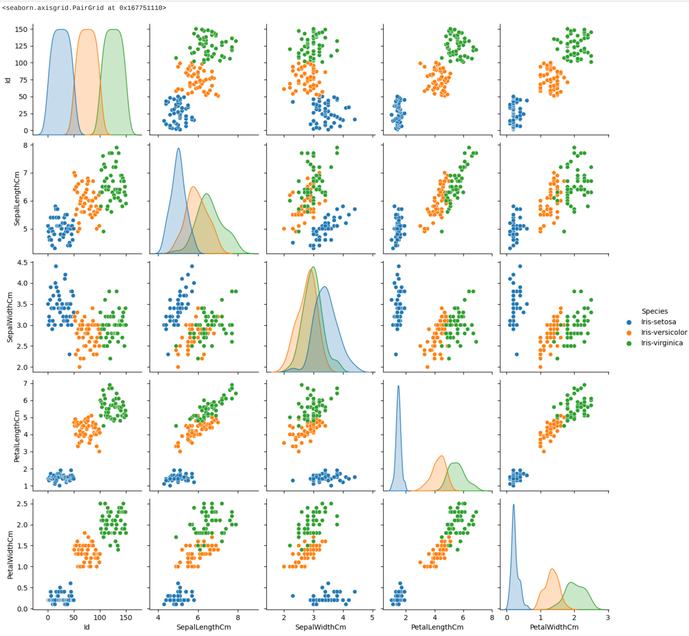
In [8]:
```
#DATA VISUALIZATION

x = plt.xticks(rotation=90)
sns.barplot(x=df['Species'],y=df['SepalWidthCm'])
```

Out[8]:
```
<Axes: xlabel='Species', ylabel='SepalWidthCm'>
```



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

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x167751110>
```



In [10]:
```
# DIVIDING THE DATA SET INTO TRAINING AND TEST DATASETS

from sklearn.model_selection import train_test_split
x = df.drop(columns=['Species'])
y = df['Species']
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.40)
```

In [11]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(90, 5)
(60, 5)
(90,)
(60,)
```

In [12]:
```
#  MACHINE LEARNING MODELS

from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
```

In [13]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
for col in x.select_dtypes(include='object'):
    x[col] = label_encoder.fit_transform(features[col])
```

In [14]:
```
from sklearn import metrics

#To train test

knn = KNeighborsClassifier(n_neighbors=3)
rfc = RandomForestClassifier(n_estimators = 7,criterion = 'entropy',random_state =7)
svc = SVC()
lc = LogisticRegression()

# making predictions on the training accuracy
for clf in (rfc, knn, svc,lc):
    clf.fit(x_train, y_train)
    label_pred = clf.predict(x_train)
    print("Accuracy score of ",clf.__class__.__name__,"=",100*metrics.accuracy_score(y_train, label_pred))
```

```
Accuracy score of  RandomForestClassifier = 100.0
Accuracy score of  KNeighborsClassifier = 98.88888888888889
Accuracy score of  SVC = 98.88888888888889
Accuracy score of  LogisticRegression = 100.0
/Users/gerishgr/anaconda3/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [15]:
```
#TO TESTING ACCURACY

for clf in (rfc, knn, svc,lc):
    clf.fit(x_train, y_train)
    label_pred = clf.predict(x_test)
    print("Accuracy score of ",clf.__class__.__name__,"=",100*metrics.accuracy_score(y_test,label_pred))
```

```
Accuracy score of  RandomForestClassifier = 98.33333333333333
Accuracy score of  KNeighborsClassifier = 98.33333333333333
Accuracy score of  SVC = 96.66666666666667
Accuracy score of  LogisticRegression = 100.0
/Users/gerishgr/anaconda3/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [16]:
```
# CONFUSION MATRIX

from sklearn.metrics import confusion_matrix

# CONFUSION MATRIX FOR TEST SET
conf_matrix_test = confusion_matrix(y_test,y_test)
print("Confusion Matrix (Test Set):")
print(conf_matrix_test)

# CONFUSION MATRIX FOR TRAINING SET
conf_matrix_train = confusion_matrix(y_train, y_train)
print("\nConfusion Matrix (Training Set):")
print(conf_matrix_train)
```

```
Confusion Matrix (Test Set):
[[22  0  0]
 [ 0 20  0]
 [ 0  0 18]]

Confusion Matrix (Training Set):
[[28  0  0]
 [ 0 30  0]
 [ 0  0 32]]
```

In [ ]: