# Importing the library

In [1]:

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

import warnings
warnings.filterwarnings('ignore')
```

# Importing the dataset

In [2]:

```
df=pd.read_csv(r"C:\Users\91956\Desktop\IRIS.csv")
df
```

Out[2]:

| | sepal_length | sepal_width | petal_length | petal_width | 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 × 5 columns

# EDA

## Shape

In [3]:

```
df.shape
```

Out[3]:

```
(150, 5)
```

## Columns and their types

In [4]:

```
df.columns
```

Out[4]:

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

In [5]:

```
df.dtypes
```

Out[5]:

```
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
dtype: object
```

In [6]:

```
df.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
```

## Null value analysis

In [7]:

```
df.isna().sum()
```

Out[7]:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

## Duplicate values

In [8]:

```
df.duplicated().sum()
```

Out[8]:

```
3
```

In [9]:

```
df[df.duplicated()]
```

Out[9]:

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **34** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| **37** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| **142** | 5.8 | 2.7 | 5.1 | 1.9 | Iris-virginica |

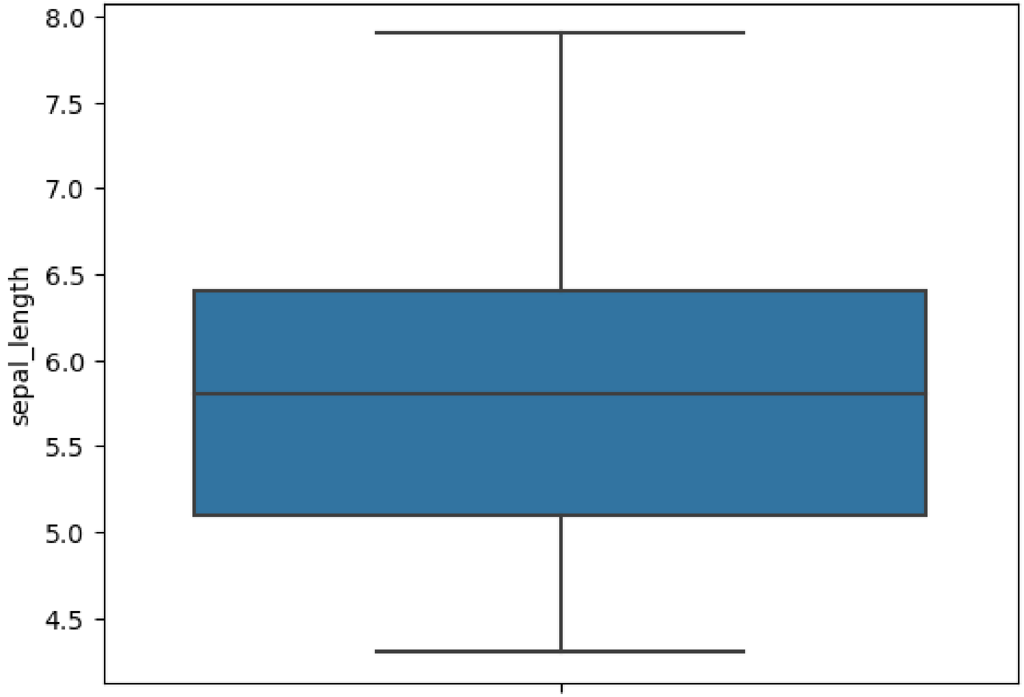
In [10]:

```
df.drop_duplicates(inplace=True)
```

## Outlier's detection

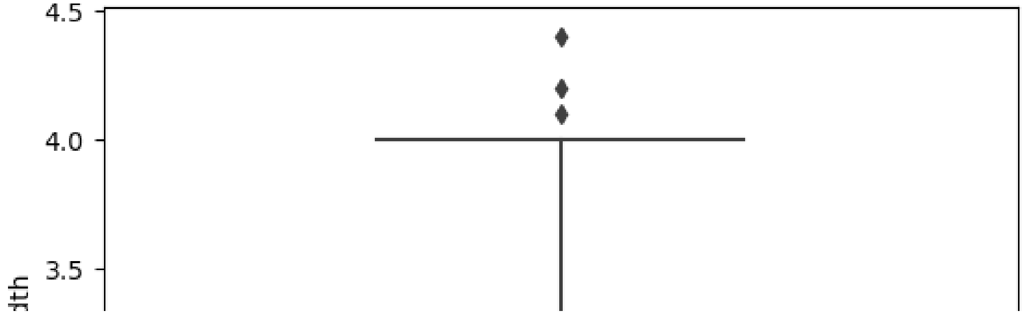
In [11]:

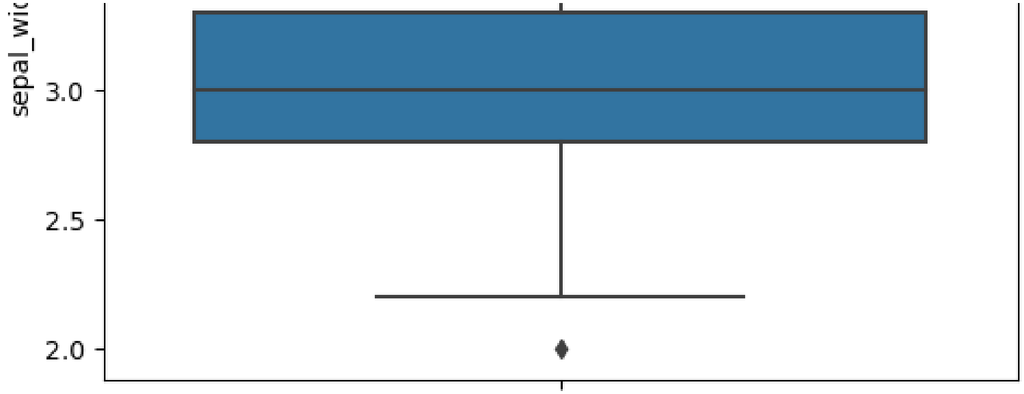
```
sns.boxplot(y='sepal_length',data=df)
plt.show()
```




In [12]:

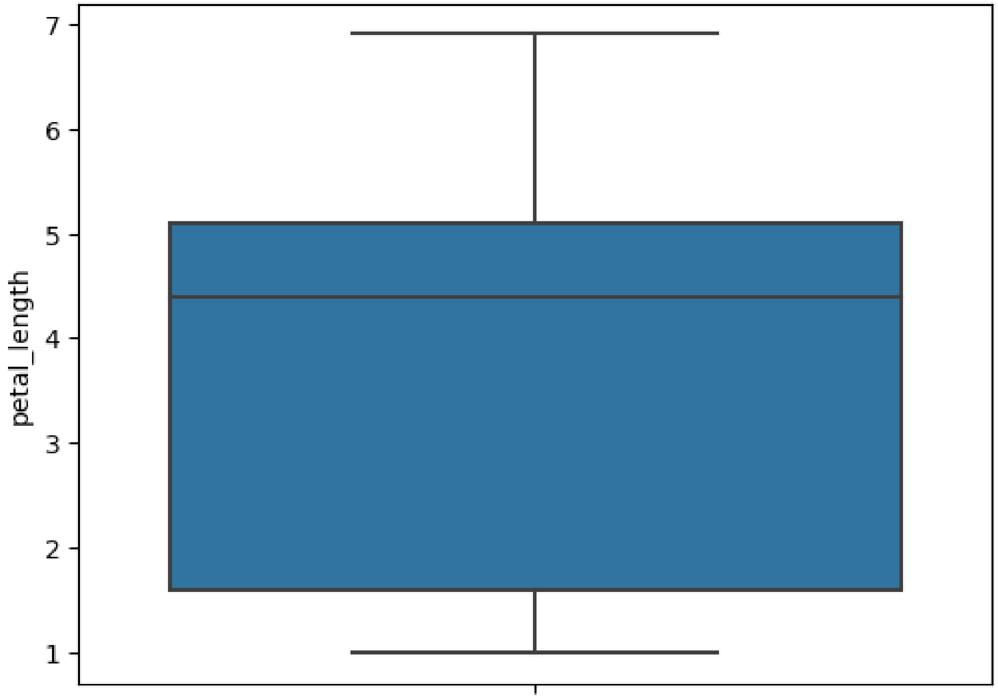
```
sns.boxplot(y='sepal_width',data=df)
plt.show()
```





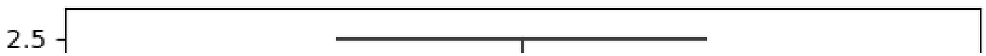

In [13]:

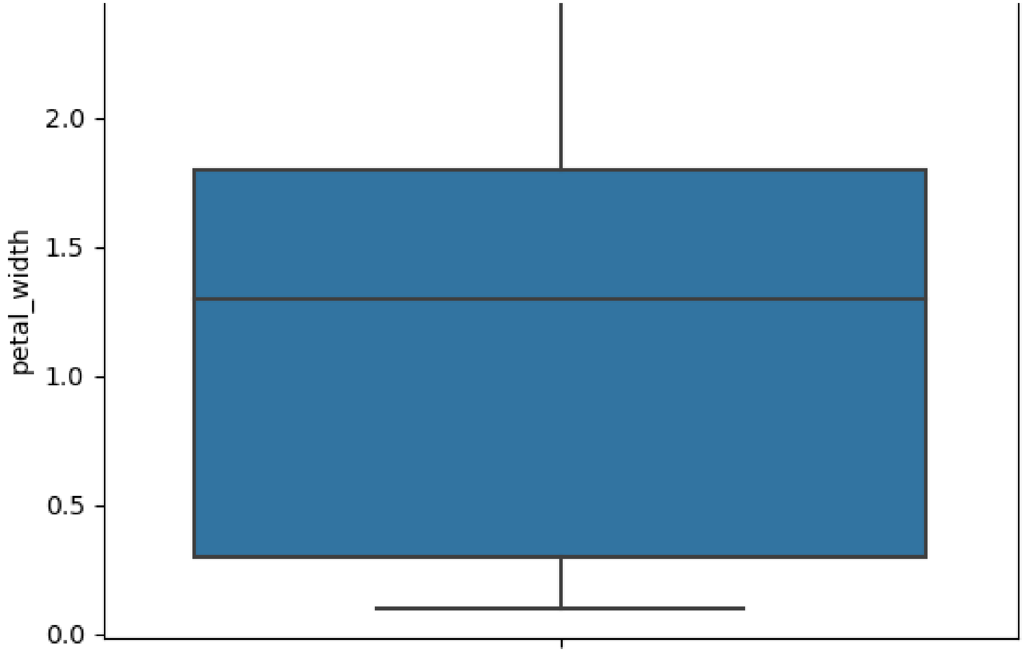
```
sns.boxplot(y='petal_length',data=df)
plt.show()
```




In [14]:

```
sns.boxplot(y='petal_width',data=df)
plt.show()
```






## Statistical summary

In [15]:

```
df.describe()
```

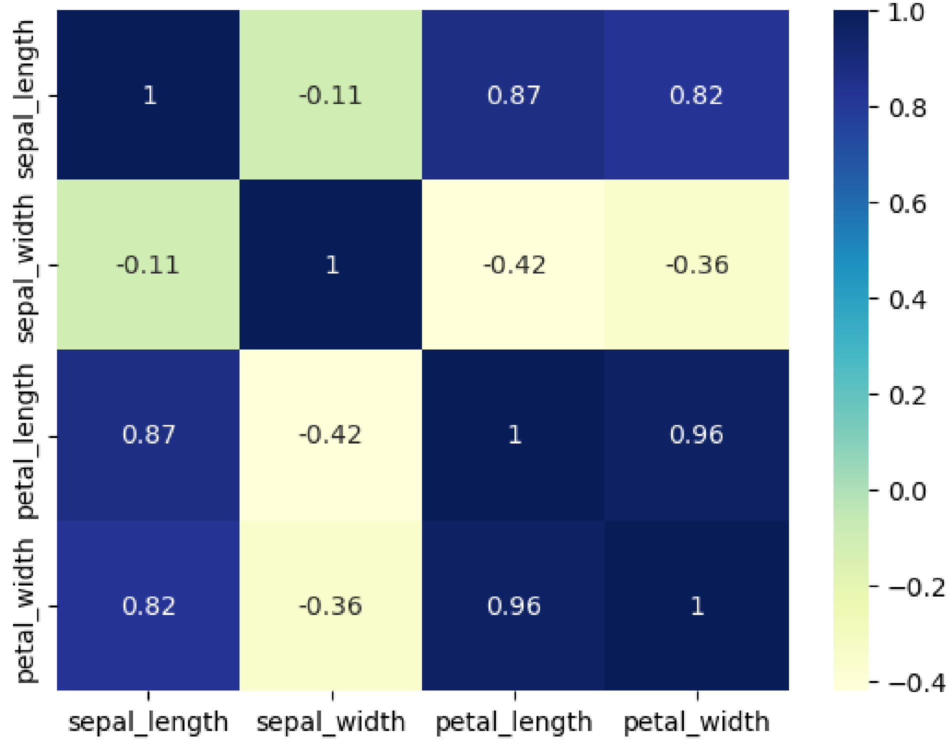
Out[15]:

| | sepal_length | sepal_width | petal_length | petal_width |
|---|---|---|---|---|
| **count** | 147.000000 | 147.000000 | 147.000000 | 147.000000 |
| **mean** | 5.856463 | 3.055782 | 3.780272 | 1.208844 |
| **std** | 0.829100 | 0.437009 | 1.759111 | 0.757874 |
| **min** | 4.300000 | 2.000000 | 1.000000 | 0.100000 |
| **25%** | 5.100000 | 2.800000 | 1.600000 | 0.300000 |
| **50%** | 5.800000 | 3.000000 | 4.400000 | 1.300000 |
| **75%** | 6.400000 | 3.300000 | 5.100000 | 1.800000 |
| **max** | 7.900000 | 4.400000 | 6.900000 | 2.500000 |

## Correlation

In [23]:

```
sns.heatmap(df.corr(), cmap = "YlGnBu", annot = True)
plt.show()
```




# Interpretation

In [16]:

```
df
```

Out[16]:

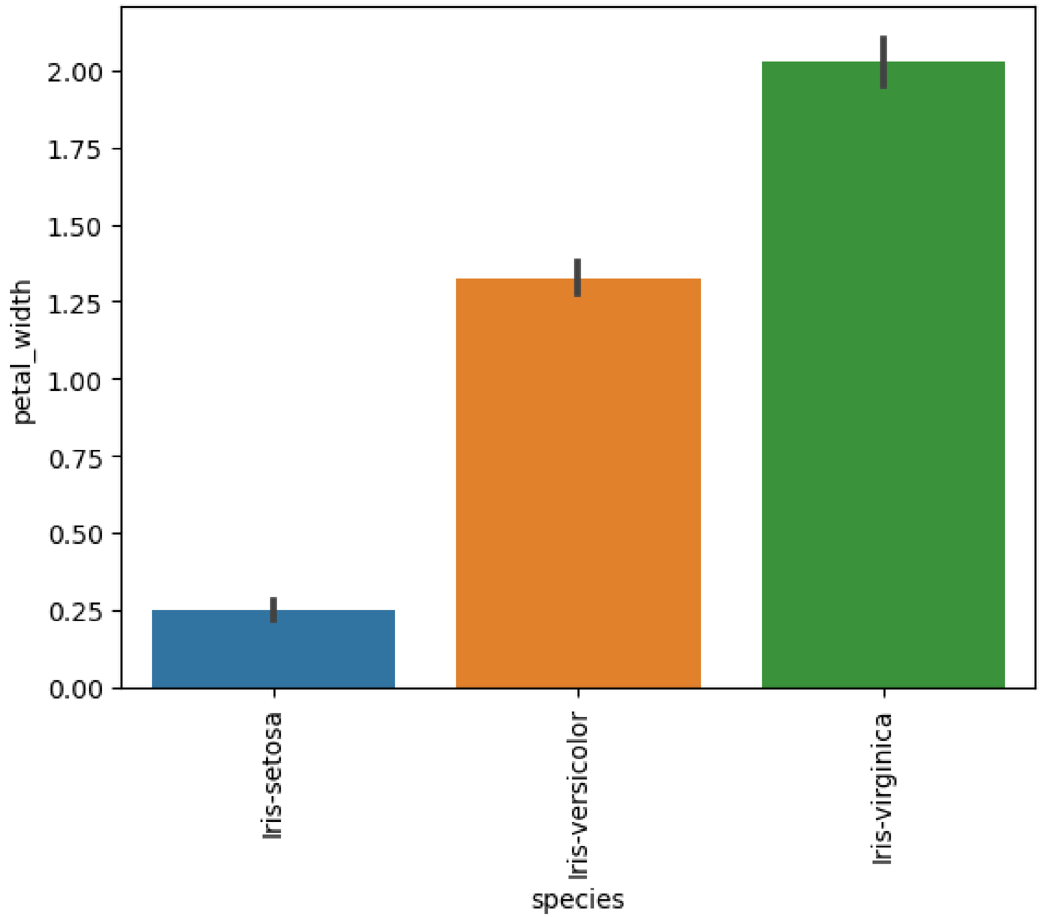
| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **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 |

**147 rows × 5 columns**

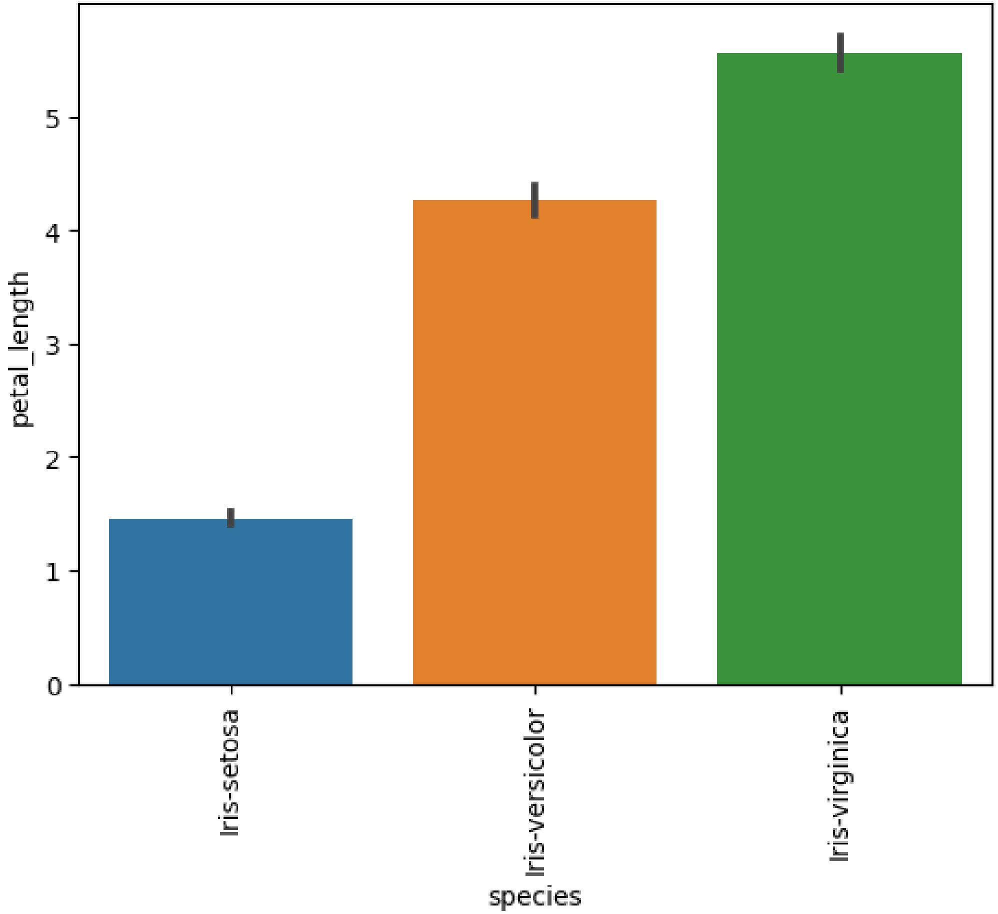
In [18]:

```
# Highest number of species according to petal_width
sns.barplot(x=df['species'],y=df['petal_width'])
plt.xticks(rotation=90)
plt.show()
```



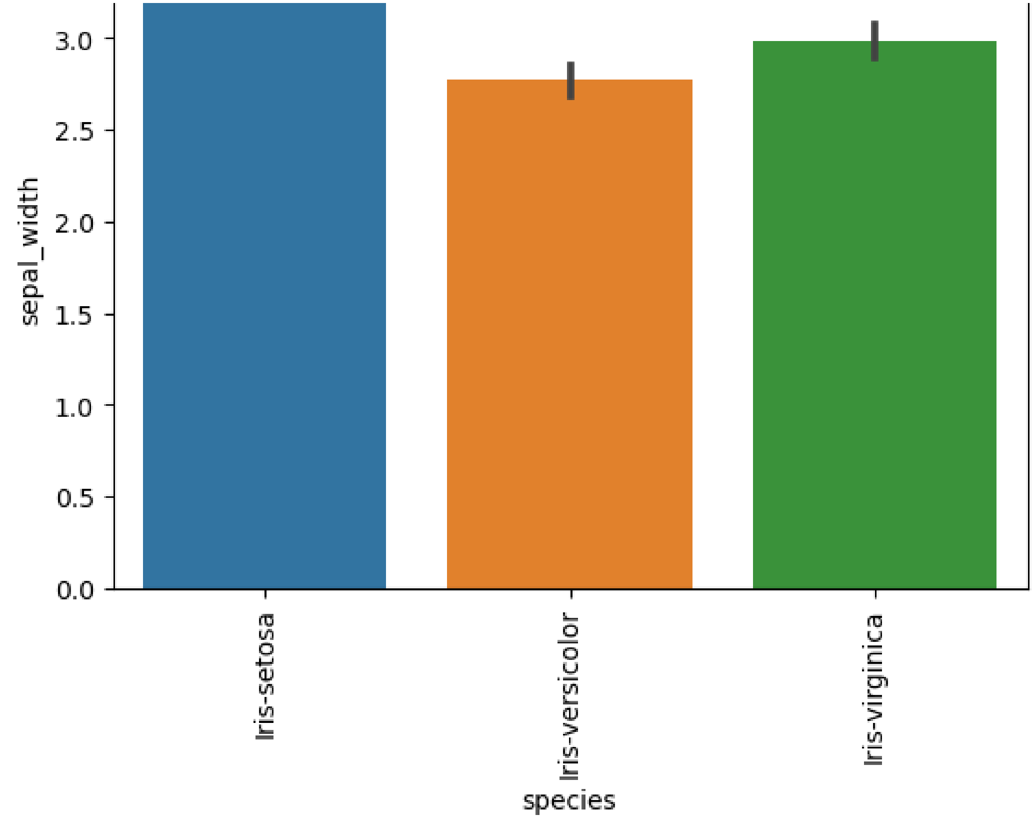
In [19]:

```
# # Highest number of species according to petal_length
sns.barplot(x=df['species'],y=df['petal_length'])
plt.xticks(rotation=90)
plt.show()
```



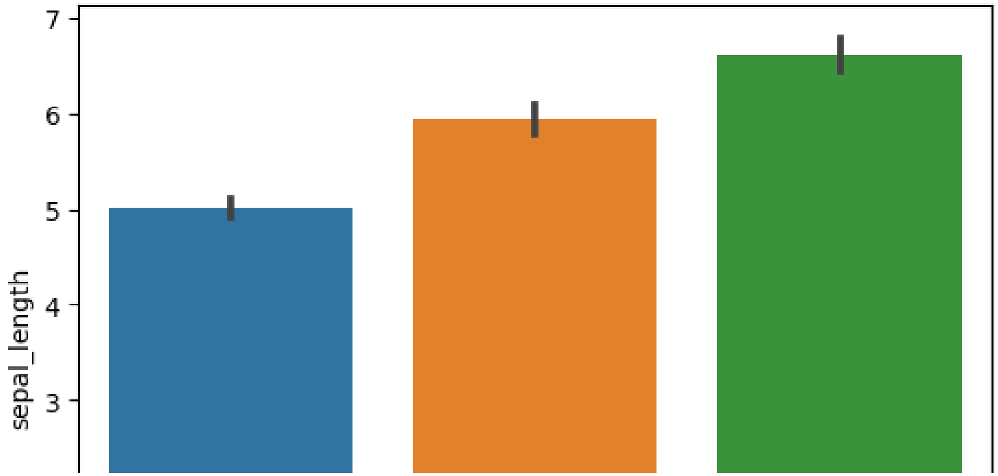

In [20]:

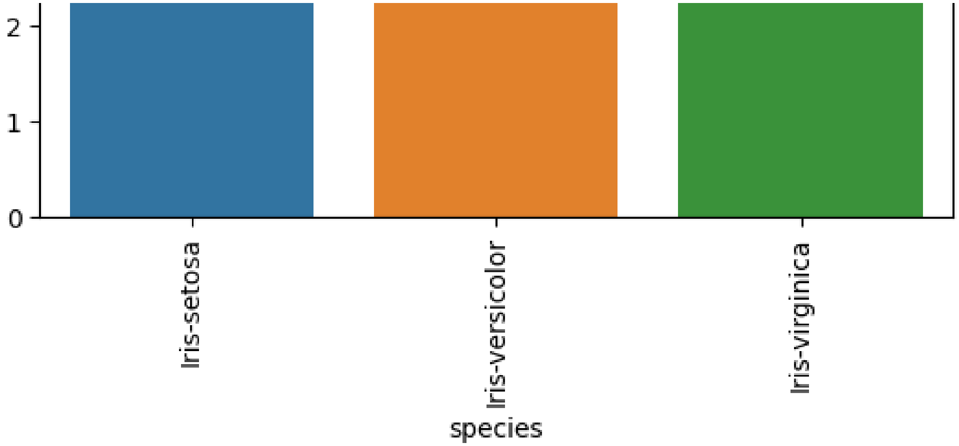
```
#Highest number of species according to sepal_width
sns.barplot(x=df['species'],y=df['sepal_width'])
plt.xticks(rotation=90)
plt.show()
```




In [21]:

```
#Highest number of species according to sepal_length
sns.barplot(x=df['species'],y=df['sepal_length'])
plt.xticks(rotation=90)
plt.show()
```






## Replacing the values of object column with numbers

In [25]:

```
df['species']=df['species'].replace({'Iris-setosa':0,'Iris-versicolor':1,'Iris-virginica':2})
```

In [24]:

```
df.species.value_counts()
```

Out[24]:

```
Iris-versicolor    50
Iris-virginica     49
Iris-setosa        48
Name: species, dtype: int64
```

# Machine learning

In [26]:

```
x=df.drop('species',axis=1)
y=df['species']
```

## Splitting the dataset into training and testing

In [16]:

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

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=7)
```

## Importing the algorithm and the performance measure

In [17]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
```

In [18]:

```
classifier=LogisticRegression()
lr=classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
y_pred_train=classifier.predict(x_train)
```

In [19]:

```
print(classification_report(y_test,y_pred))
print(classification_report(y_train,y_pred_train))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       0.83      0.83      0.83        12
           2       0.82      0.82      0.82        11

    accuracy                           0.87        30
   macro avg       0.88      0.88      0.88        30
weighted avg       0.87      0.87      0.87        30

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       1.00      0.97      0.99        38
           2       0.97      1.00      0.99        39

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```

In [20]:

```
from sklearn.tree import DecisionTreeClassifier
classifier2=DecisionTreeClassifier(random_state=7, max_depth=5,
                                   criterion='gini',max_leaf_nodes=7)
```

```
dt=classifier2.fit(x_train,y_train)
y_pred=classifier2.predict(x_test)
y_pred_train=classifier2.predict(x_train)

print(classification_report(y_test,y_pred))
print(classification_report(y_train,y_pred_train))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       0.91      0.83      0.87        12
           2       0.83      0.91      0.87        11

    accuracy                           0.90        30
   macro avg       0.91      0.91      0.91        30
weighted avg       0.90      0.90      0.90        30

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       1.00      1.00      1.00        38
           2       1.00      1.00      1.00        39

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```

In [21]:

```
from sklearn.ensemble import RandomForestClassifier
classifier4=RandomForestClassifier(random_state=7, max_depth=7,
                                   criterion='gini',max_leaf_nodes=15,
                                  min_samples_split=40,n_estimators=30)

rf=classifier4.fit(x_train,y_train)
y_pred=classifier4.predict(x_test)
y_pred_train=classifier4.predict(x_train)

print(classification_report(y_test,y_pred))
print(classification_report(y_train,y_pred_train))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       0.83      0.83      0.83        12
           2       0.82      0.82      0.82        11

    accuracy                           0.87        30
   macro avg       0.88      0.88      0.88        30
weighted avg       0.87      0.87      0.87        30
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       0.95      1.00      0.97        38
           2       1.00      0.95      0.97        39

    accuracy                           0.98       120
   macro avg       0.98      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```

In [22]:

```
from sklearn.svm import SVC
svc=SVC(random_state=7)
svm=svc.fit(x_train,y_train)
y_pred=svc.predict(x_test)
y_pred_train=svc.predict(x_train)

print(classification_report(y_test,y_pred))
print(classification_report(y_train,y_pred_train))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       0.83      0.83      0.83        12
           2       0.82      0.82      0.82        11

    accuracy                           0.87        30
   macro avg       0.88      0.88      0.88        30
weighted avg       0.87      0.87      0.87        30

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       1.00      0.97      0.99        38
           2       0.97      1.00      0.99        39

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```

In [23]:

```
from sklearn.ensemble import AdaBoostClassifier
adbc=AdaBoostClassifier(random_state=7)
adbc1=adbc.fit(x_train,y_train)
y_pred=adbc.predict(x_test)
```

```
y_pred_train=adbc.predict(x_train)

print(classification_report(y_test,y_pred))
print(classification_report(y_train,y_pred_train))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         7
           1       0.83      0.83      0.83        12
           2       0.82      0.82      0.82        11

    accuracy                           0.87        30
   macro avg       0.88      0.88      0.88        30
weighted avg       0.87      0.87      0.87        30

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       0.95      1.00      0.97        38
           2       1.00      0.95      0.97        39

    accuracy                           0.98       120
   macro avg       0.98      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```