In [2]:

```
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score

sns.set_style('darkgrid')
```

In [3]:

```
## name of datasets that are present in seaborn library
sns.get_dataset_names()
```

```
Out[3]: ['anagrams',
         'anscombe',
         'attention',
         'brain_networks',
         'car_crashes',
         'diamonds',
         'dots',
         'dowjones',
         'exercise',
         'flights',
         'fmri',
         'geyser',
         'glue',
         'healthexp',
         'iris',
         'mpg',
         'penguins',
         'planets',
         'seaice',
         'taxis',
         'tips',
         'titanic',
         'anagrams',
         'anagrams',
         'anscombe',
         'anscombe',
         'attention',
         'attention',
         'brain_networks',
         'brain_networks',
         'car_crashes',
         'car_crashes',
         'diamonds',
         'diamonds',
         'dots',
         'dots',
         'dowjones',
         'dowjones',
         'exercise',
         'exercise',
         'flights',
         'flights',
         'fmri',
         'fmri',
         'geyser',
         'geyser',
         'glue',
         'glue',
         'healthexp',
         'healthexp',
         'iris',
         'iris',
         'mpg',
         'mpg',
         'penguins',
         'penguins',
         'planets',
 'planets',
 'seaice',
 'seaice',
 'taxis',
 'taxis',
 'tips',
 'tips',
 'titanic',
 'titanic',
 'anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

In [24]:

```
## iris dataset
df = sns.load_dataset('iris')
```

In [25]:

```
df.head()
```

Out[25]:

	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 [26]:

```
df.shape
```

Out[26]:

```
(150, 5)
```

In [27]:

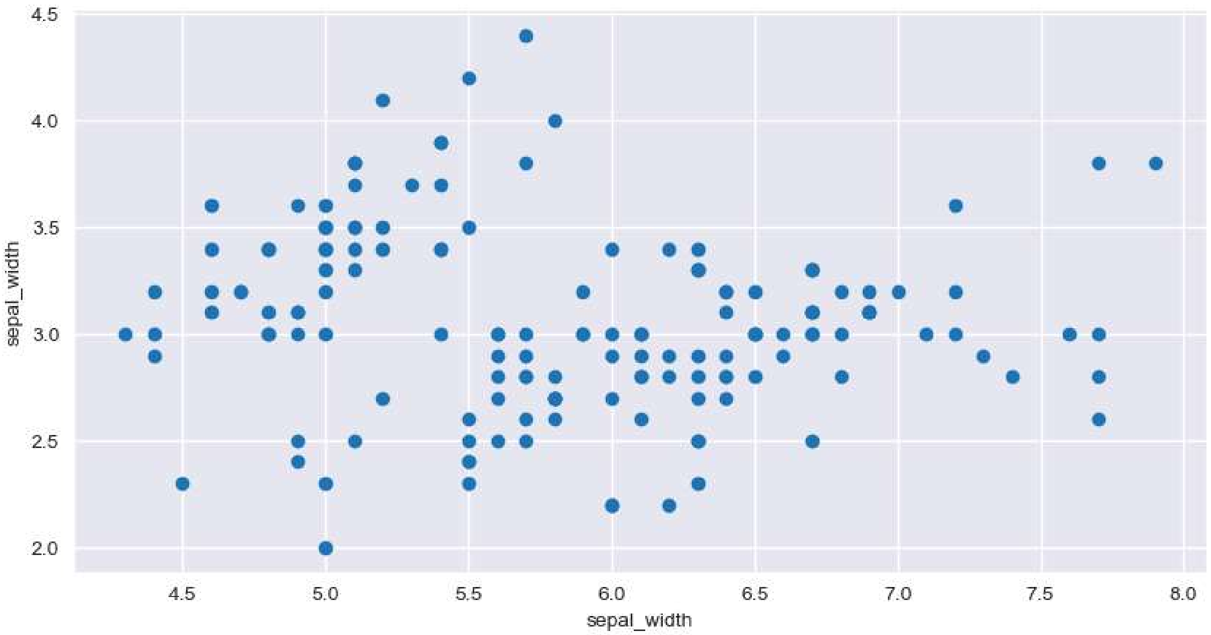
```
## count the species present in species column
df.species.value_counts()
```

Out[27]:

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

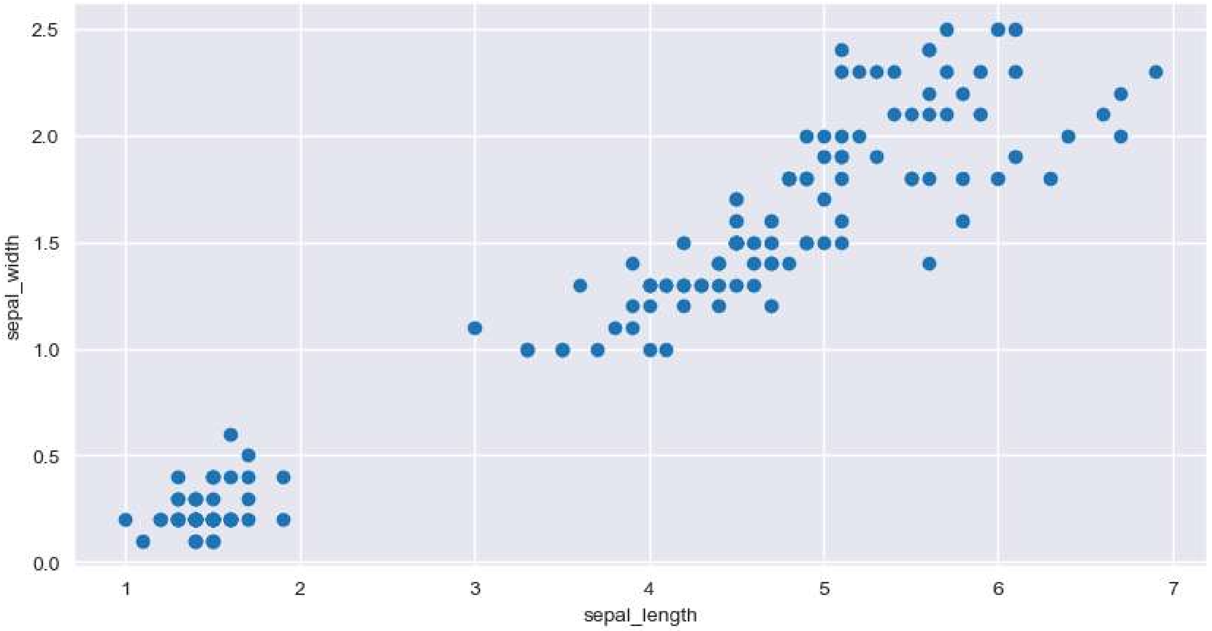
In [28]:

```
plt.figure(figsize=(10,5))
plt.scatter(df["sepal_length"],df['sepal_width']);
plt.ylabel('sepal_width')
plt.xlabel('sepal_width');
```



In [29]:

```
plt.figure(figsize=(10, 5))
plt.scatter(df['petal_length'], df['petal_width'], marker='o');
plt.ylabel('sepal_width')
plt.xlabel('sepal_length');
```



In [30]:

```
## checking the null values
df.isnull().sum()
```

Out[30]:

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

In [31]:

```
df.species.unique()
```

Out[31]:

```
array(['setosa', 'versicolor', 'virginica'], dtype=object)
```

In [32]:

```
## converting species into label encoding
def map_species(f):
    if f == 'setosa':
        f=0
    elif f== 'versicolor':
        f=1
    elif f== 'virginica':
        f=2
    return f
```

In [33]:

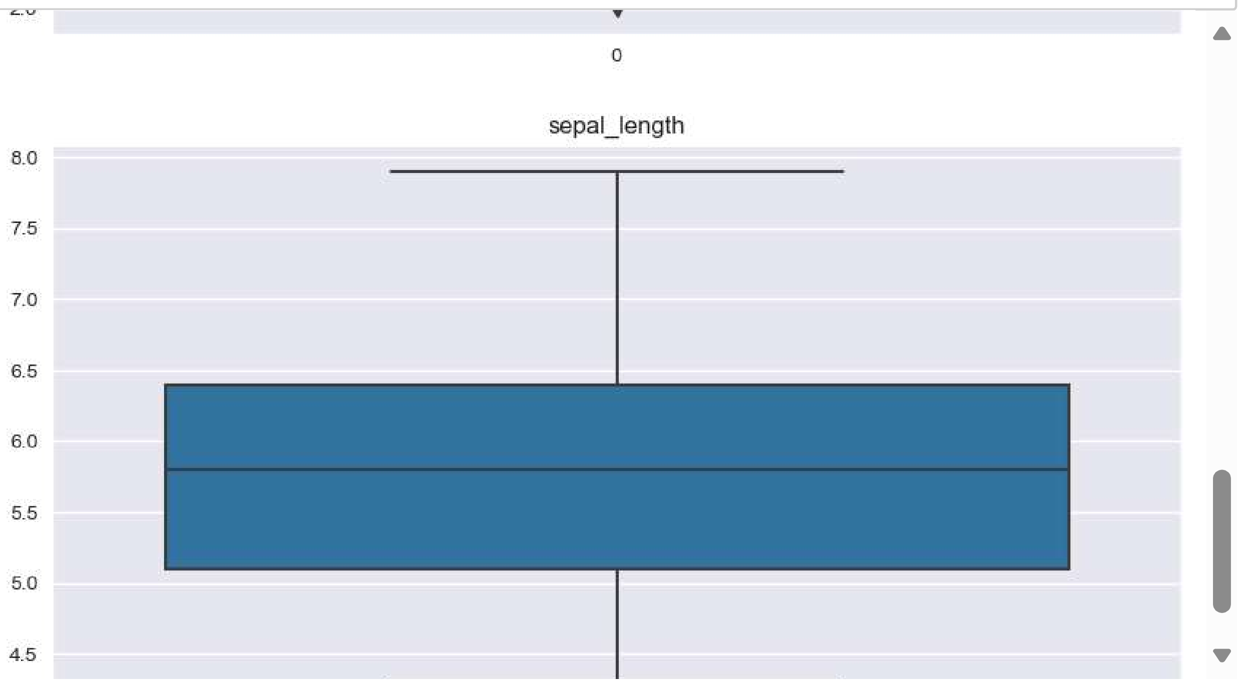
```
df['species'] = df.species.map(map_species)
```

In [37]:
```
df.head()
```

Out[37]:

	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

In [39]:
```
for f in  ['sepal_length', 'petal_length', 'sepal_width', 'sepal_length']:
    plt.figure(figsize=(10,5))
    sns.boxplot(df[f])
    plt.title(f)
```



In [40]:
```
## independent and dependent features
X=df.iloc[:, :-1].values
y=df.iloc[:,-1].values
```

In [41]:
```
## preprocessing
sc=StandardScaler()
X=sc.fit_transform(X)
```

In [44]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25, random_state
```

In [45]:
```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

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

In [46]:
```
lg=LogisticRegression()
```

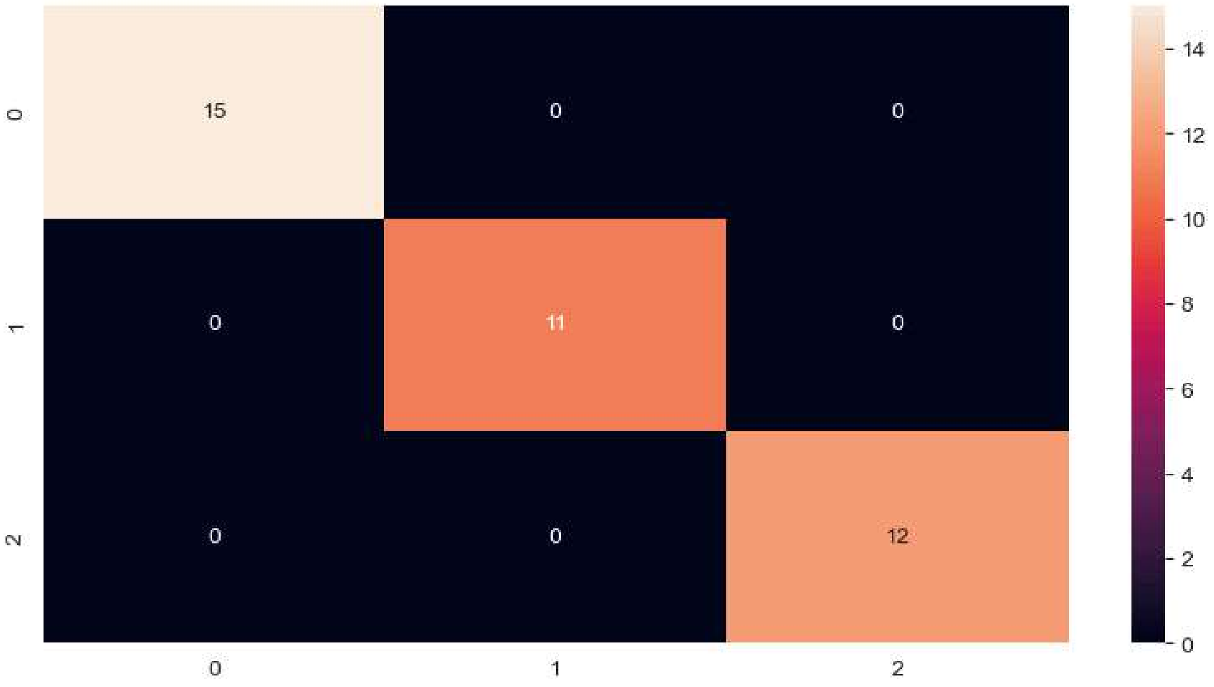
In [47]:
```
lg.fit(X_train,y_train)
```

Out[47]:
```
▾  LogisticRegression ⓘ ?
                        (https://scikit-
                        learn.org/1.4/modules/generated/sklearn.linear_model.LogisticRegres
LogisticRegression()
```

In [48]:
```
pred=lg.predict(X_test)
```

```
In [49]: print("-------------------------------------------Classification Report---------
         print(classification_report(y_test, pred))

         print("-----------------------------------------Accuracy Score-----------------
         print(accuracy_score(y_test, pred))

         print("-------------------------------------------Confustion Matrix------------
         plt.figure(figsize=(10,5))
         sns.heatmap(confusion_matrix(y_test, pred), annot=True);
```

```
--------------------------------------------Classification Report---------------
------------------------------
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      1.00      1.00        11
           2       1.00      1.00      1.00        12

    accuracy                           1.00        38
   macro avg       1.00      1.00      1.00        38
weighted avg       1.00      1.00      1.00        38

------------------------------------------Accuracy Score------------------------
---------------------------------
1.0
--------------------------------------------Confustion Matrix-------------------
----------------------------------
```



```
In [50]: pd.DataFrame({'Actual': y_test,  'Predicted': pred}).head(50)
```

Out[50]:

	Actual	Predicted
0	1	1
1	0	0
2	2	2
3	1	1
4	1	1
5	0	0
6	1	1
7	2	2
8	1	1
9	1	1
10	2	2
11	0	0
12	0	0
13	0	0
14	0	0
15	1	1
16	2	2
17	1	1
18	1	1
19	2	2
20	0	0
21	2	2
22	0	0
23	2	2
24	2	2
25	2	2
26	2	2
27	2	2
28	0	0
29	0	0
30	0	0
31	0	0
32	1	1
33	0	0
34	0	0
35	2	2

	Actual	Predicted
36	1	1
37	0	0

```
In [ ]:
```