In [1]:
```python
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.datasets import load_iris
```

In [2]:
```python
iris=load_iris()
```

In [3]:
```python
features=iris.data
```

In [4]:
```python
target=iris.target
```

In [5]:
```python
df=pd.DataFrame(features,columns=['sepal length in cm', 'sepal width in cm' ,'petal length in cm' ,'petal width in cm'])
```

In [6]:
```python
df['target']=target
```

In [7]:
```python
df.head()
```

Out[7]:

	sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	target
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 [8]:
```python
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 in cm  150 non-null    float64
 1   sepal width in cm   150 non-null    float64
 2   petal length in cm  150 non-null    float64
 3   petal width in cm   150 non-null    float64
 4   target              150 non-null    int32  
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```

In [9]:
```python
df.describe()
```

Out[9]:

	sepal length in cm	sepal width in cm	petal length in cm	petal width in cm	target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

In [10]:
```python
df.columns
```

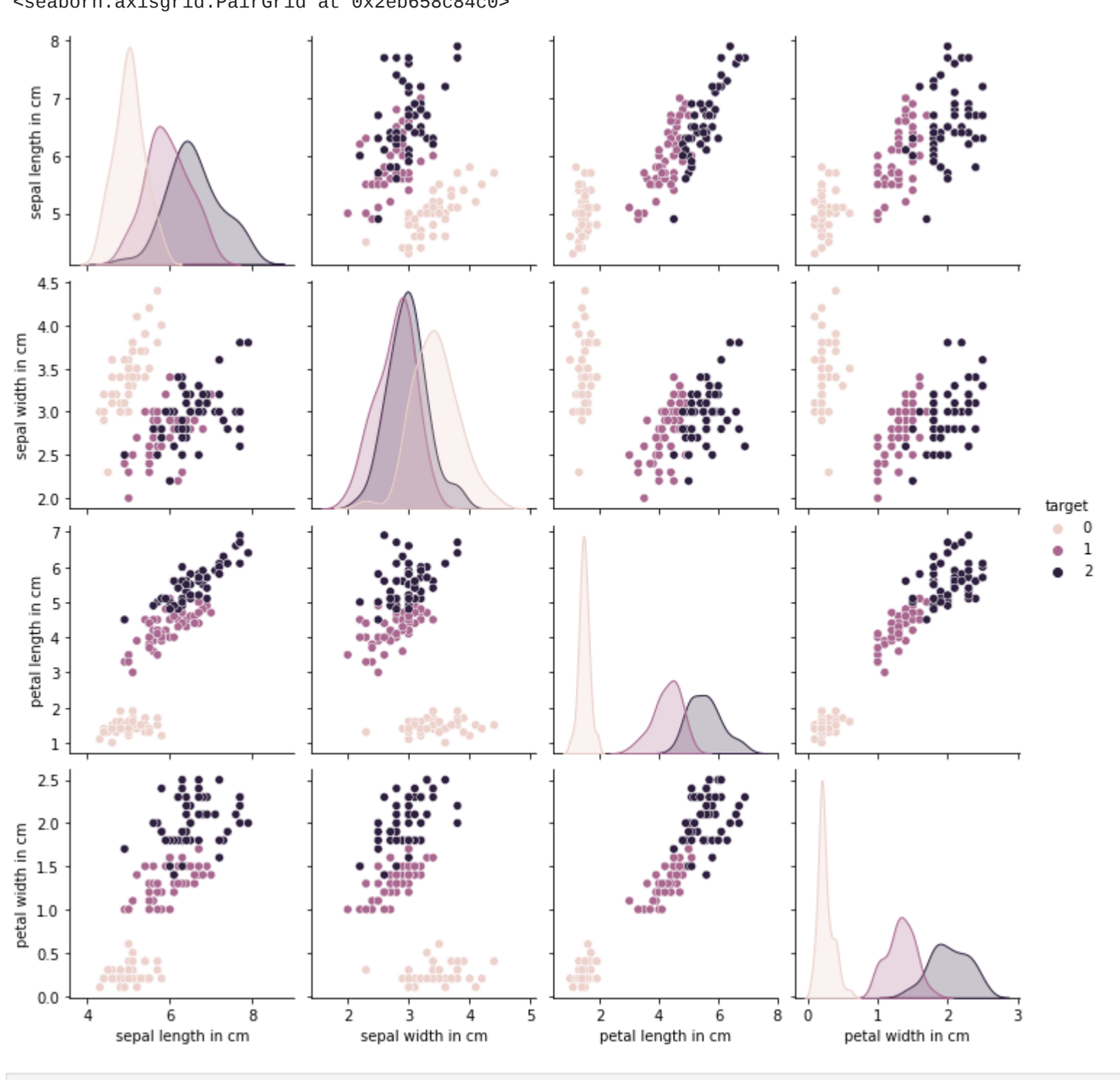
Out[10]:
```
Index(['sepal length in cm', 'sepal width in cm', 'petal length in cm',
       'petal width in cm', 'target'],
      dtype='object')
```

In [11]:
```python
df['target'].value_counts()
```

Out[11]:
```
0    50
1    50
2    50
Name: target, dtype: int64
```

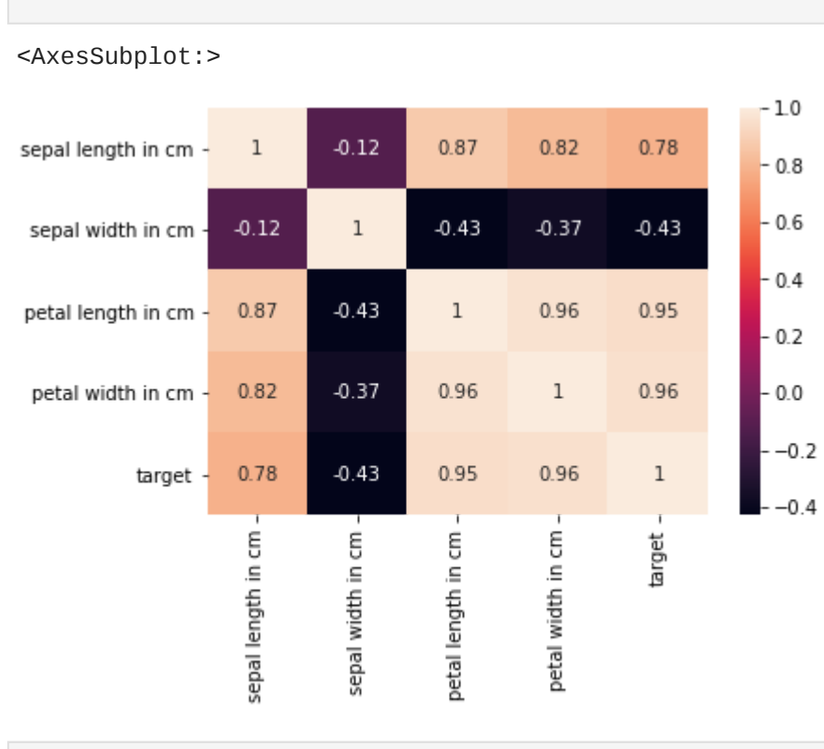
In [12]:
```python
sns.pairplot(df,hue='target')
```

Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x2eb658c84c0>
```



In [13]:
```python
sns.heatmap(df.corr(),annot=True)
```

Out[13]:
```
<AxesSubplot:>
```



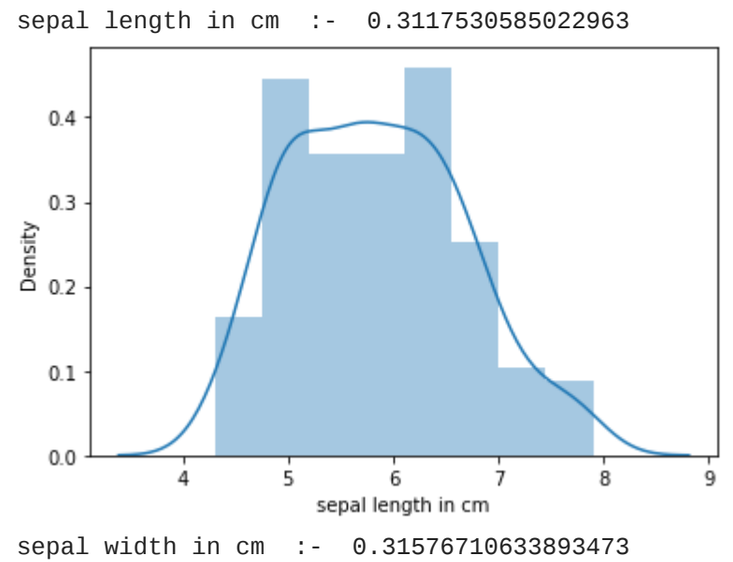
In [14]:
```python
df.skew()
```

Out[14]:
```
sepal length in cm    0.314911
sepal width in cm     0.318966
petal length in cm   -0.274884
petal width in cm    -0.102967
target                0.000000
dtype: float64
```

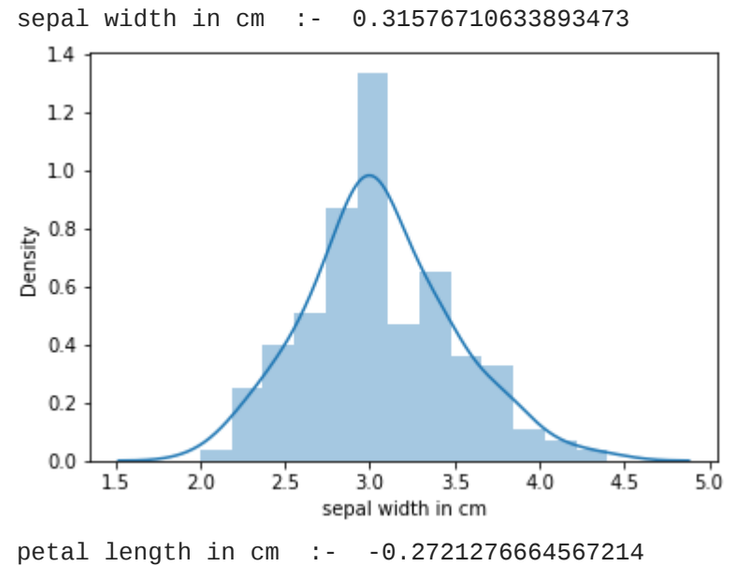
In [15]:
```python
from scipy.stats import skew
```

In [16]:
```python
for col in df:

    print(col,  ' :-  ',end='')
    print(skew(df[col]))


    sns.distplot(df[col])
    plt.show()
```

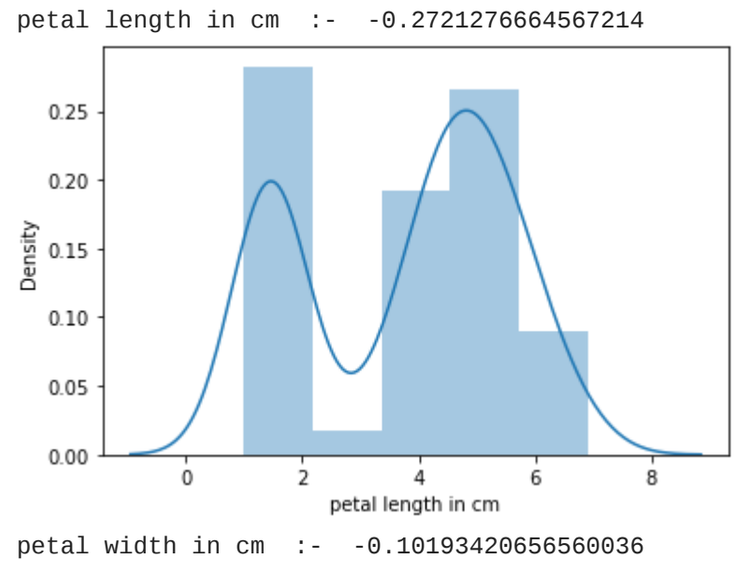
```
sepal length in cm  :-  0.3117530585022963
```



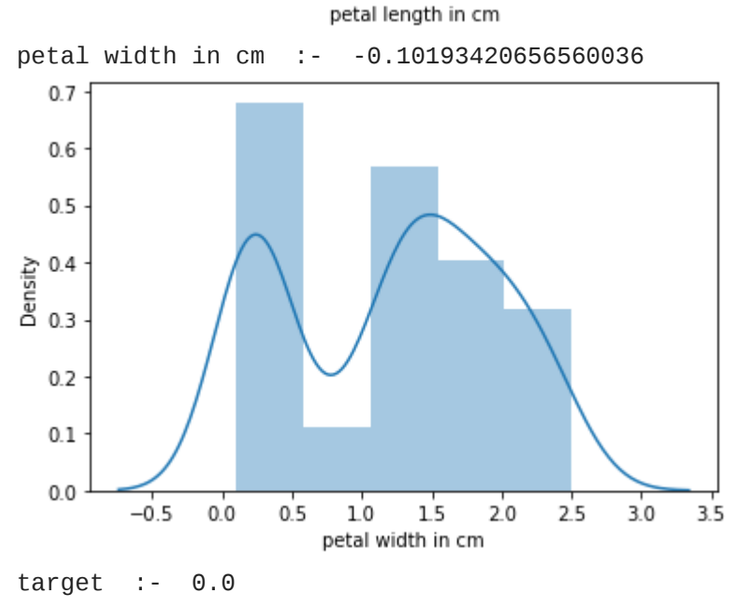

```
sepal width in cm  :-  0.31576710633893473
```



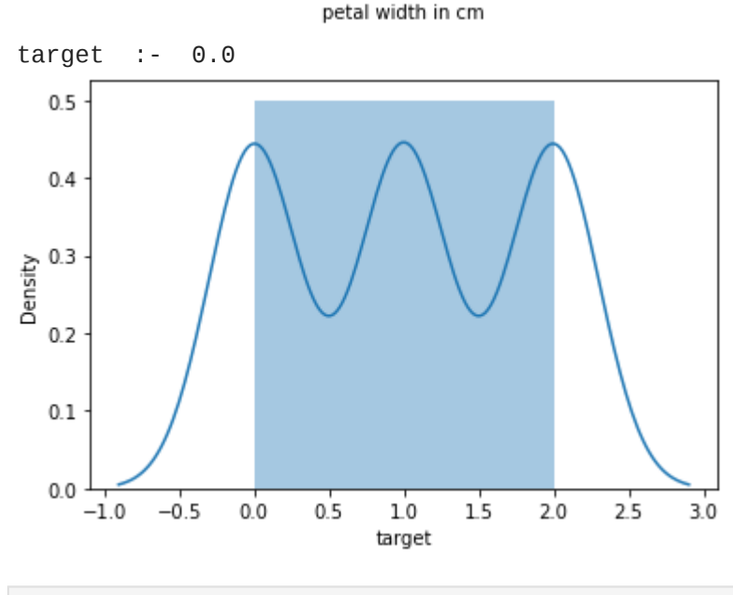

```
petal length in cm  :-  -0.2721276664567214
```



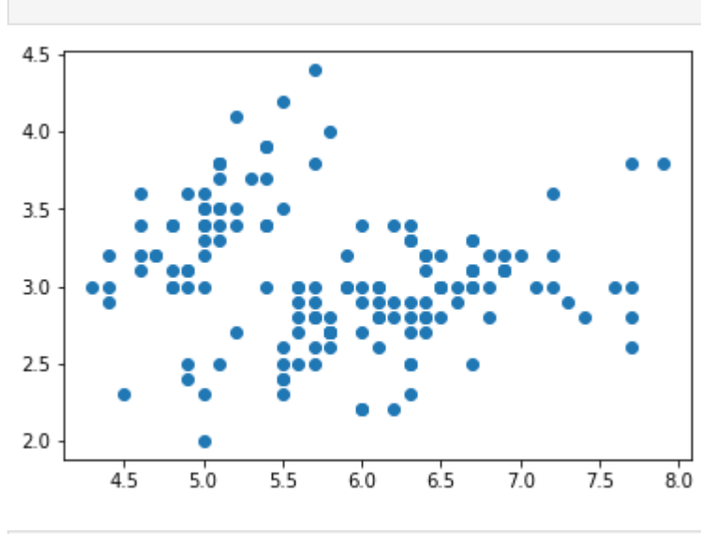

```
petal width in cm  :-  -0.10193420656560036
```



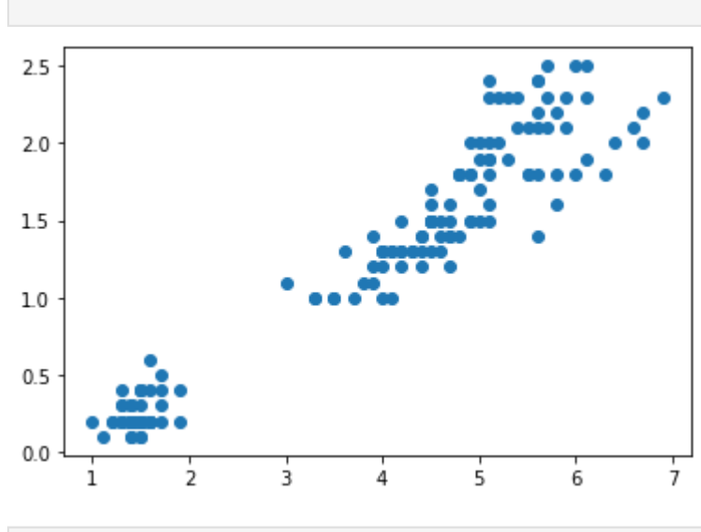

```
target  :-  0.0
```



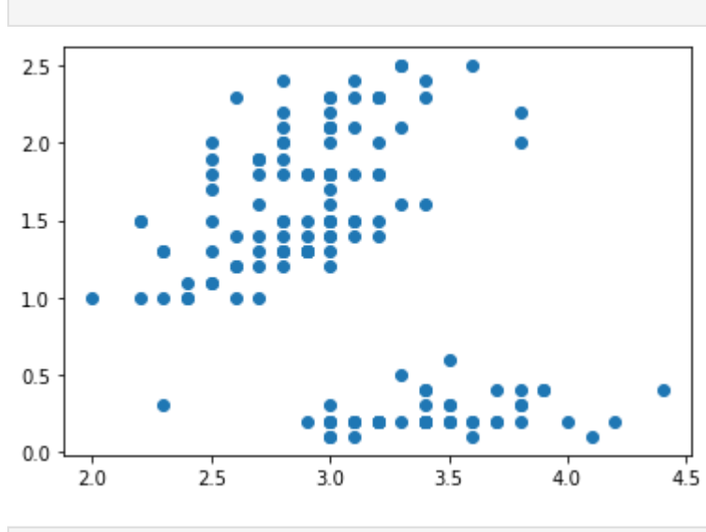
In [17]:
```python
plt.scatter(data=df,x='sepal length in cm',y='sepal width in cm')
plt.show()
```



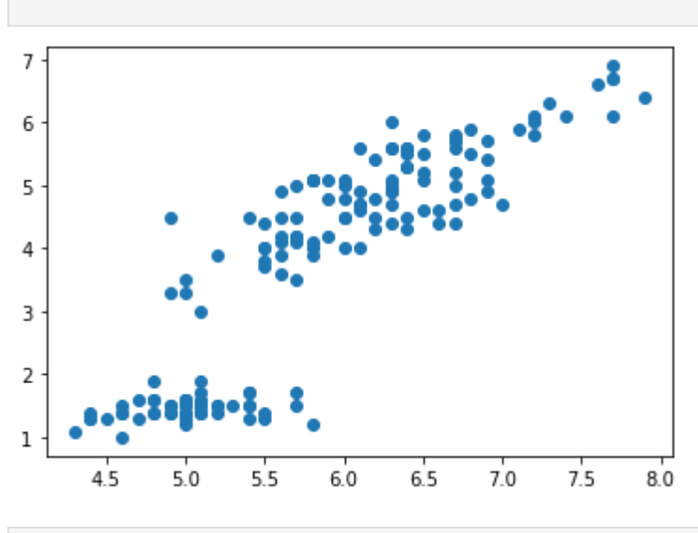
In [18]:
```python
plt.scatter(data=df,x='petal length in cm',y='petal width in cm')
plt.show()
```



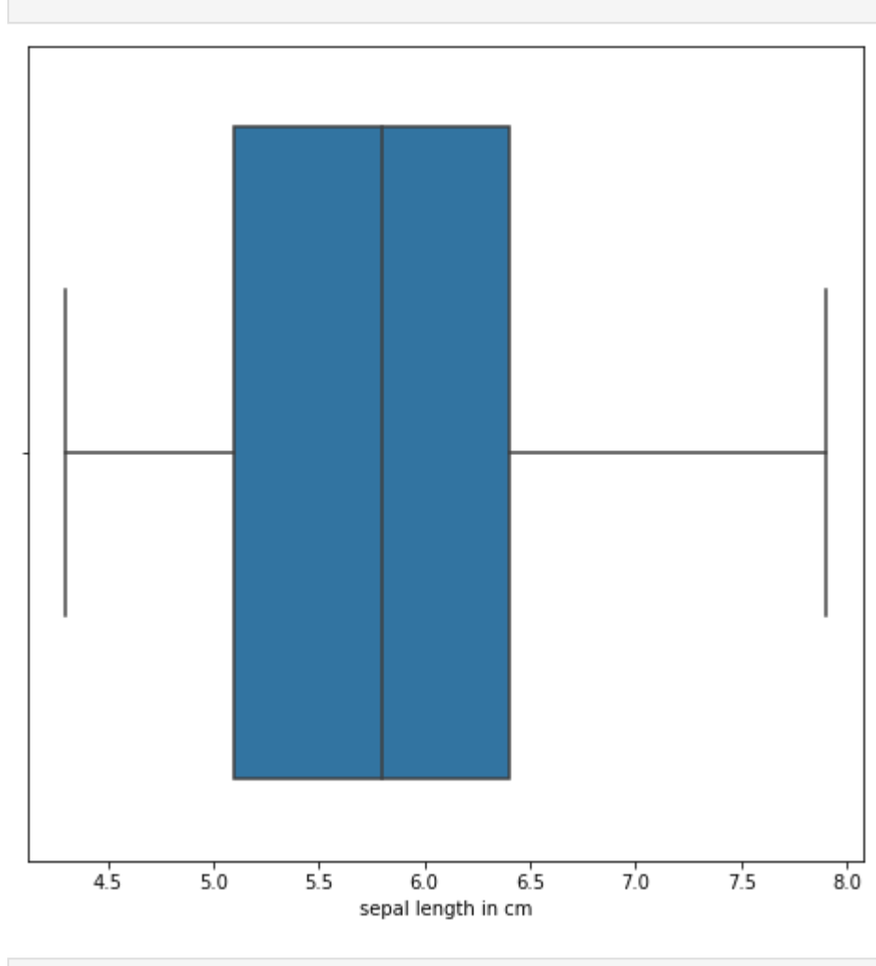
In [19]:
```python
plt.scatter(data=df,x='sepal width in cm',y='petal width in cm')
plt.show()
```



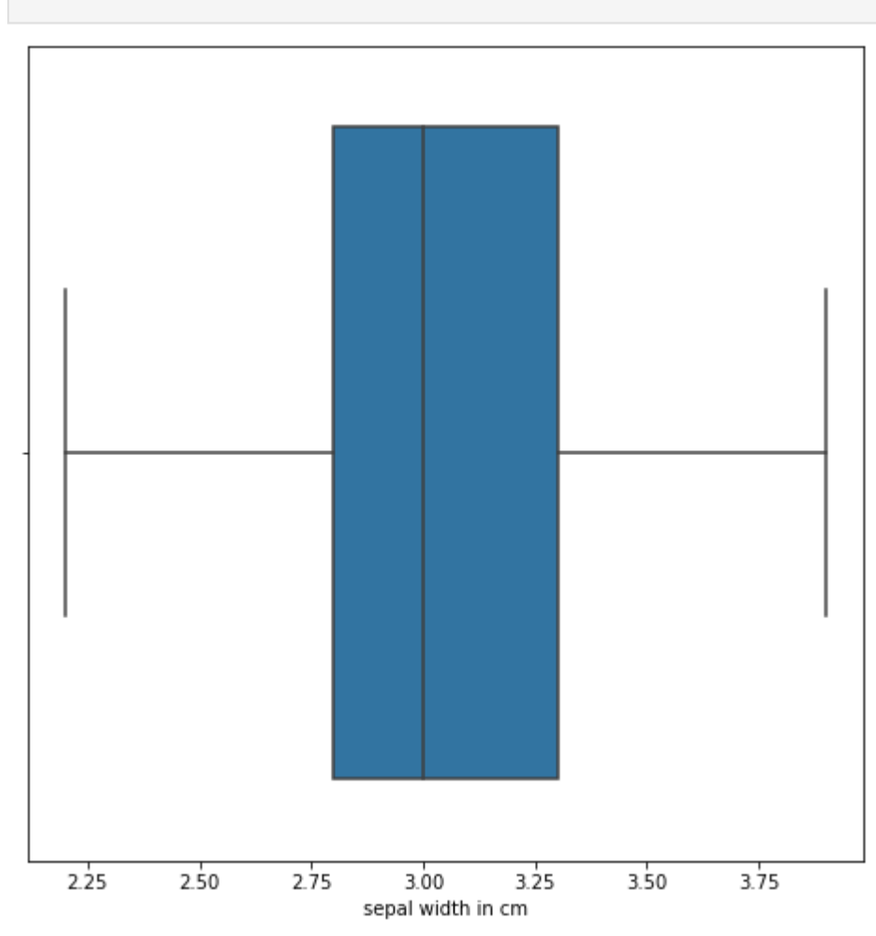
In [20]:
```python
plt.scatter(data=df,x='sepal length in cm',y='petal length in cm')
plt.show()
```



In [60]:
```python
plt.figure(figsize=(8,8))
sns.boxplot(data=df,x='sepal length in cm',hue='target')
plt.show()
```

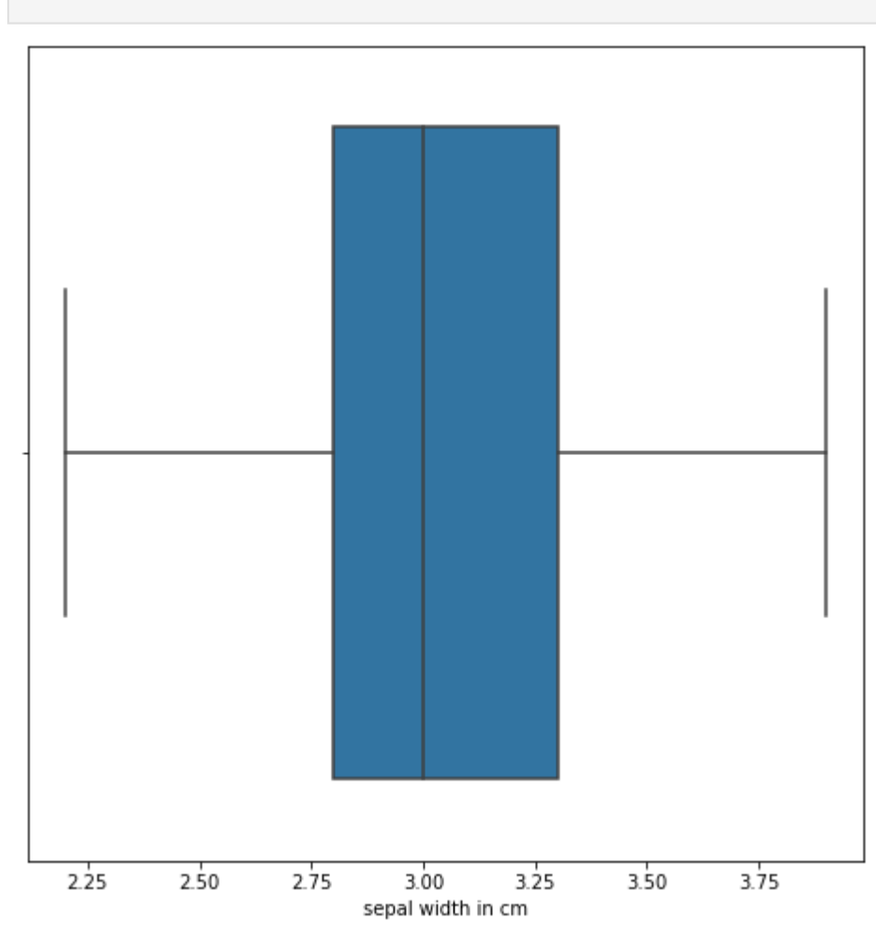


In [70]:
```python
plt.figure(figsize=(8,8))
sns.boxplot(data=df,x='sepal width in cm',hue='target')
plt.show()
```

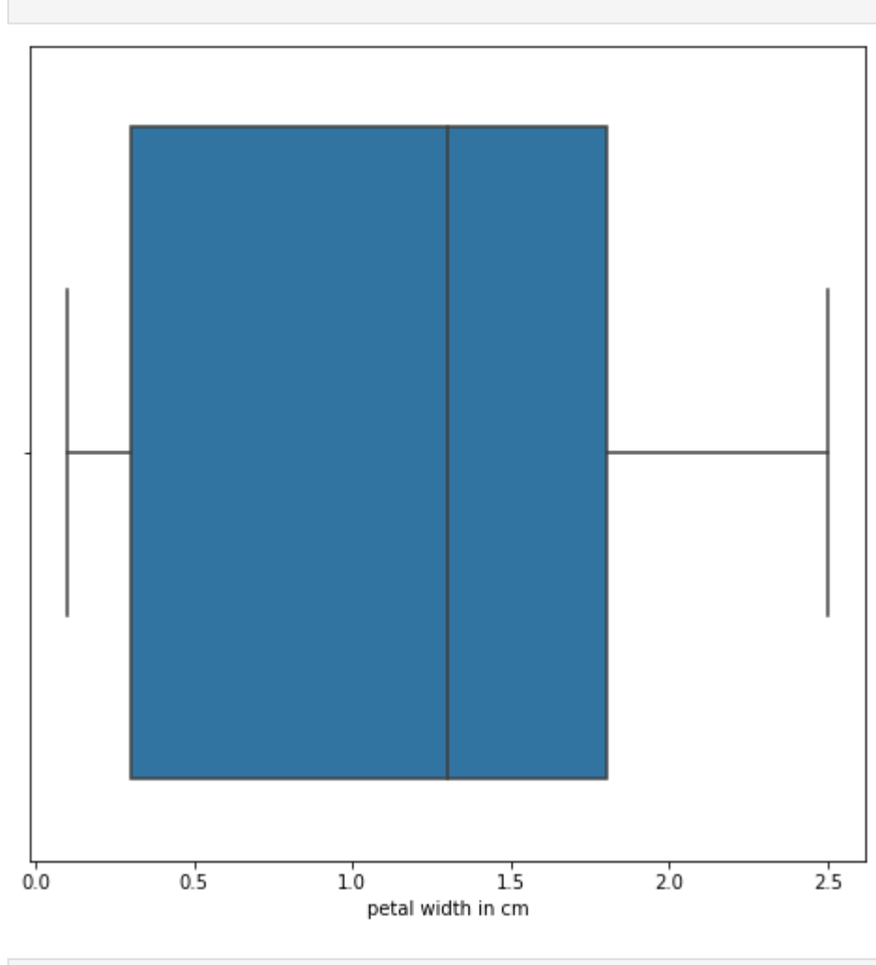


In [71]:
```python
df.drop(df[df['sepal width in cm']<=2.1].index,axis=0,inplace=True)
df.drop(df[df['sepal width in cm']>=4.0].index,axis=0,inplace=True)
```

In [72]:
```python
plt.figure(figsize=(8,8))
sns.boxplot(data=df,x='sepal width in cm',hue='target')
plt.show()
```

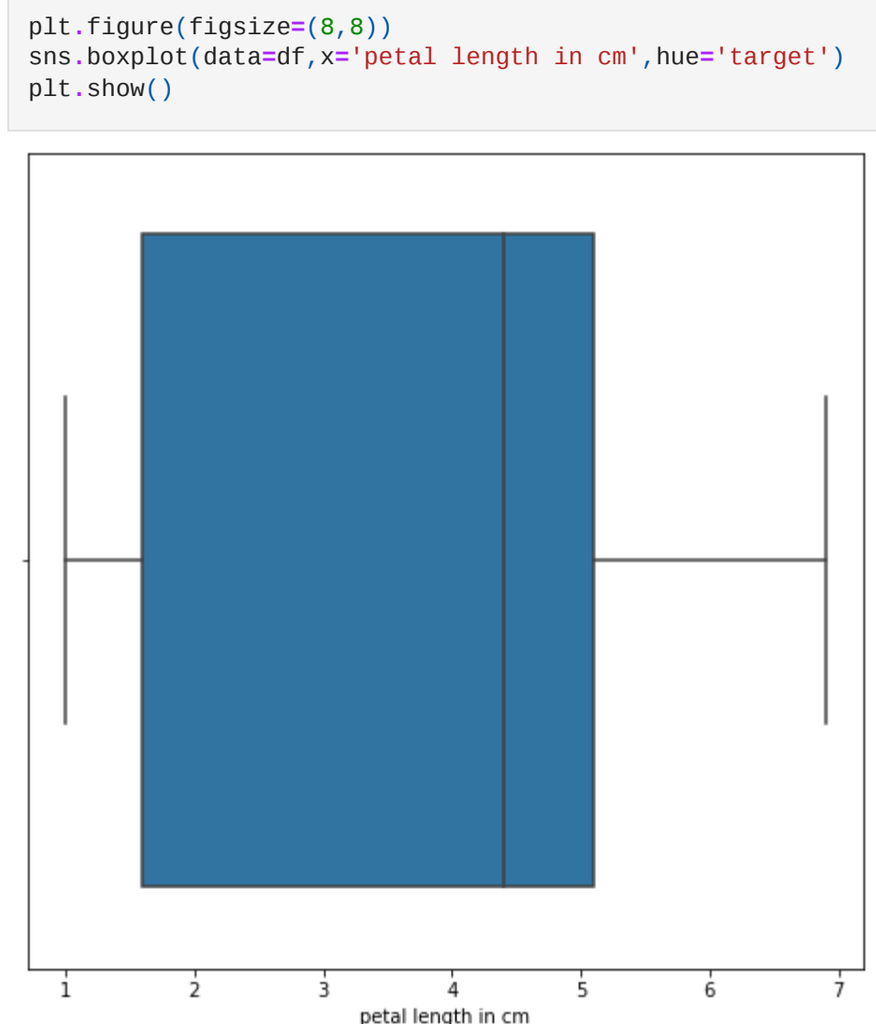


In [73]:
```python
plt.figure(figsize=(8,8))
sns.boxplot(data=df,x='petal width in cm',hue='target')
plt.show()
```



In [ ]:

In [74]:
```python
plt.figure(figsize=(8,8))
sns.boxplot(data=df,x='petal length in cm',hue='target')
plt.show()
```



In [75]:
```python
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report
from sklearn.model_selection import train_test_split ,cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier , AdaBoostClassifier   ,GradientBoostingClassifier
from xgboost import XGBClassifier
```

In [76]:
```python
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
```

In [77]:
```python
xtrain,xtest,ytrain,ytest=train_test_split(x,y,random_state=0)
```

In [78]:
```python
models=[('logistic Regression         :- ',LogisticRegression()),
        ('KNeighbors Classifier       :- ' ,KNeighborsClassifier()),
        ('Deceision Tree              :- ',DecisionTreeClassifier()),
        ('Support vectore Classifier  :- ',SVC()),
        ('Random Forest Clasifier     :- ',RandomForestClassifier()),
        ('Ada Boosting Classifier     :- ',AdaBoostClassifier()),
        ('Gradient Boosting Classifier :- ' ,GradientBoostingClassifier()),
        ('Xtreame Boosting Classifier  :- ',XGBClassifier())

       ]
```

In [83]:
```python
accuracy=[]
for name,model in models:
    print(name)
    model.fit(xtrain,ytrain)
    ypred=model.predict(xtest)
    print(classification_report(ytest,ypred))
    accuracy.append(accuracy_score(ytest,ypred))
```

```
logistic Regression         :- 
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      0.91      0.95        11
           2       0.92      1.00      0.96        11

    accuracy                           0.97        37
   macro avg       0.97      0.97      0.97        37
weighted avg       0.98      0.97      0.97        37

KNeighbors Classifier       :- 
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      0.91      0.95        11
           2       0.92      1.00      0.96        11

    accuracy                           0.97        37
   macro avg       0.97      0.97      0.97        37
weighted avg       0.98      0.97      0.97        37

Deceision Tree              :- 
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      0.91      0.95        11
           2       0.92      1.00      0.96        11

    accuracy                           0.97        37
   macro avg       0.97      0.97      0.97        37
weighted avg       0.98      0.97      0.97        37

Support vectore Classifier  :- 
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.91      0.91      0.91        11
           2       0.91      0.91      0.91        11

    accuracy                           0.95        37
   macro avg       0.94      0.94      0.94        37
weighted avg       0.95      0.95      0.95        37

Random Forest Clasifier     :- 
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      0.91      0.95        11
           2       0.92      1.00      0.96        11

    accuracy                           0.97        37
   macro avg       0.97      0.97      0.97        37
weighted avg       0.98      0.97      0.97        37

Ada Boosting Classifier     :- 
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.91      0.91      0.91        11
           2       0.91      0.91      0.91        11

    accuracy                           0.95        37
   macro avg       0.94      0.94      0.94        37
weighted avg       0.95      0.95      0.95        37

Gradient Boosting Classifier :- 
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      0.91      0.95        11
           2       0.92      1.00      0.96        11

    accuracy                           0.97        37
   macro avg       0.97      0.97      0.97        37
weighted avg       0.98      0.97      0.97        37

Xtreame Boosting Classifier  :- 
[21:18:34] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      0.91      0.95        11
           2       0.92      1.00      0.96        11

    accuracy                           0.97        37
   macro avg       0.97      0.97      0.97        37
weighted avg       0.98      0.97      0.97        37
```

In [ ]:
```python
print('mean accuracy 'np.mean(accuracy))
```

In [110…
```python
xgb=XGBClassifier(booster='gbtree',random_state=52,gamma=0.0111,reg_alpha =4.001)
xgb.fit(xtrain,ytrain)
ypred=xgb.predict(xtest)
print(classification_report(ytest,ypred))
print(accuracy_score(ytest,ypred))
```

```
[21:28:32] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.1/src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      0.91      0.95        11
           2       0.92      1.00      0.96        11

    accuracy                           0.97        37
   macro avg       0.97      0.97      0.97        37
weighted avg       0.98      0.97      0.97        37

0.972972972972973
trainnig score 0.9722222222222222
testing score  0.972972972972973
```

In [111…
```python
print('trainnig score',xgb.score(xtrain,ytrain))
print('testing score ',xgb.score(xtest,ytest))
```

```
trainnig score 0.9722222222222222
testing score  0.972972972972973
```

In [113…
```python
df.to_csv('iris_dataset')
```

In [ ]:

In [ ]:

In [ ]: