```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from sklearn.datasets import load_iris
```

```
iris=load_iris()
```

```
iris.keys()
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR',
'feature_names', 'filename', 'data_module'])
```

```
X=iris.data
y=iris.target
```

```
df=pd.DataFrame(X,columns=iris.feature_names)
```

```
df
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal
width (cm)
0                  5.1               3.5                1.4
0.2
1                  4.9               3.0                1.4
0.2
2                  4.7               3.2                1.3
0.2
3                  4.6               3.1                1.5
0.2
4                  5.0               3.6                1.4
0.2
..                 ...               ...                ...
...
145                6.7               3.0                5.2
2.3
146                6.3               2.5                5.0
1.9
147                6.5               3.0                5.2
2.0
148                6.2               3.4                5.4
2.3
149                5.9               3.0                5.1
1.8

[150 rows x 4 columns]
```

```
df['target']=y
```

```
df
```

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal
width (cm)  \
0                  5.1               3.5                1.4
0.2
1                  4.9               3.0                1.4
0.2
2                  4.7               3.2                1.3
0.2
3                  4.6               3.1                1.5
0.2
4                  5.0               3.6                1.4
0.2
..                 ...               ...                ...
...
145                6.7               3.0                5.2
2.3
146                6.3               2.5                5.0
1.9
147                6.5               3.0                5.2
2.0
148                6.2               3.4                5.4
2.3
149                5.9               3.0                5.1
1.8

     target
0         0
1         0
2         0
3         0
4         0
..      ...
145       2
146       2
147       2
148       2
149       2

[150 rows x 5 columns]
```

```
df['target'].unique()
```

```
array([0, 1, 2])
```

```
yn=iris.target_names
```

```
yn
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

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

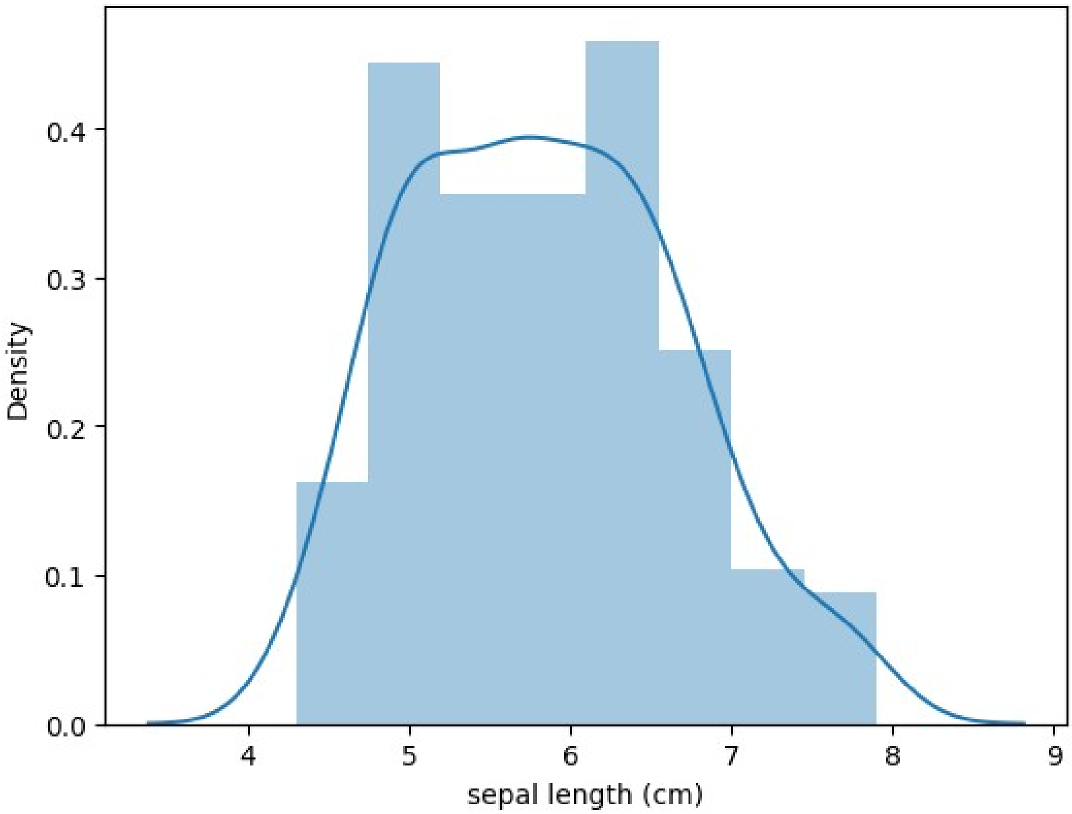
```
df.describe()
```

```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000   
mean            5.843333          3.057333           3.758000   
std             0.828066          0.435866           1.765298   
min             4.300000          2.000000           1.000000   
25%             5.100000          2.800000           1.600000   
50%             5.800000          3.000000           4.350000   
75%             6.400000          3.300000           5.100000   
max             7.900000          4.400000           6.900000   

       petal width (cm)      target  
count        150.000000  150.000000  
mean           1.199333    1.000000  
std            0.762238    0.819232  
min            0.100000    0.000000  
25%            0.300000    0.000000  
50%            1.300000    1.000000  
75%            1.800000    2.000000  
max            2.500000    2.000000  
```

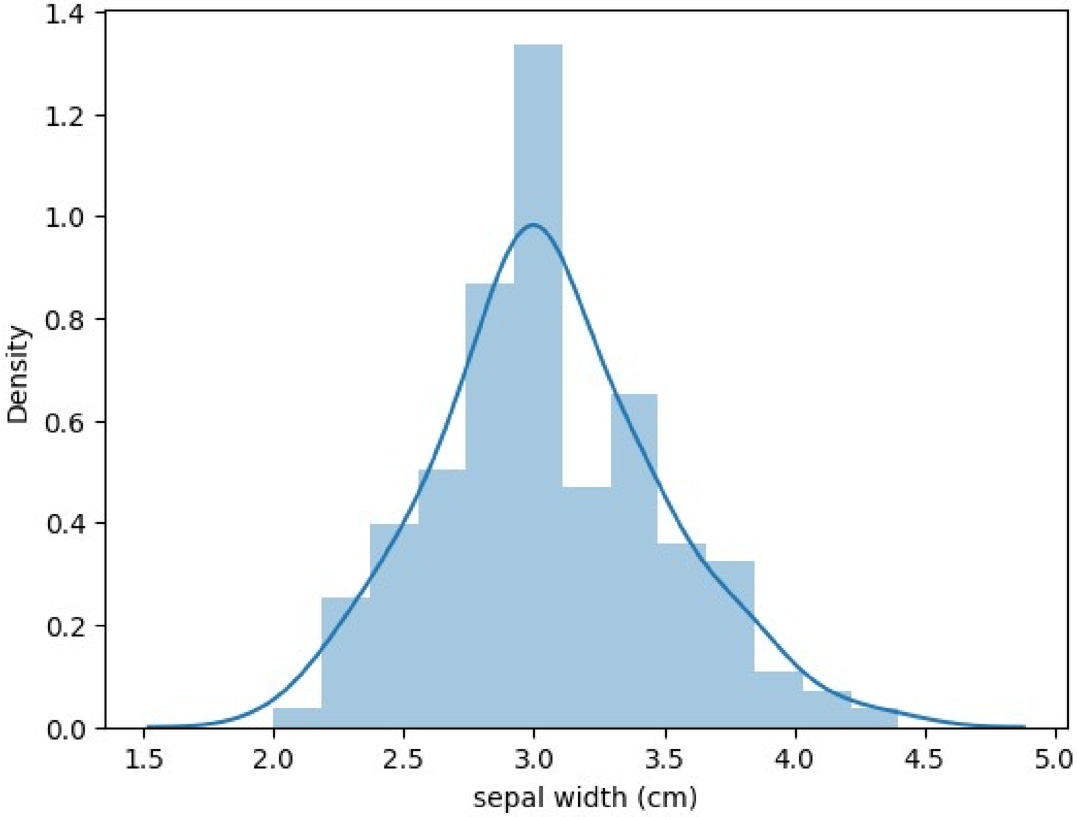
```
sns.distplot(df['sepal length (cm)'])
```

```
<AxesSubplot:xlabel='sepal length (cm)', ylabel='Density'>
```



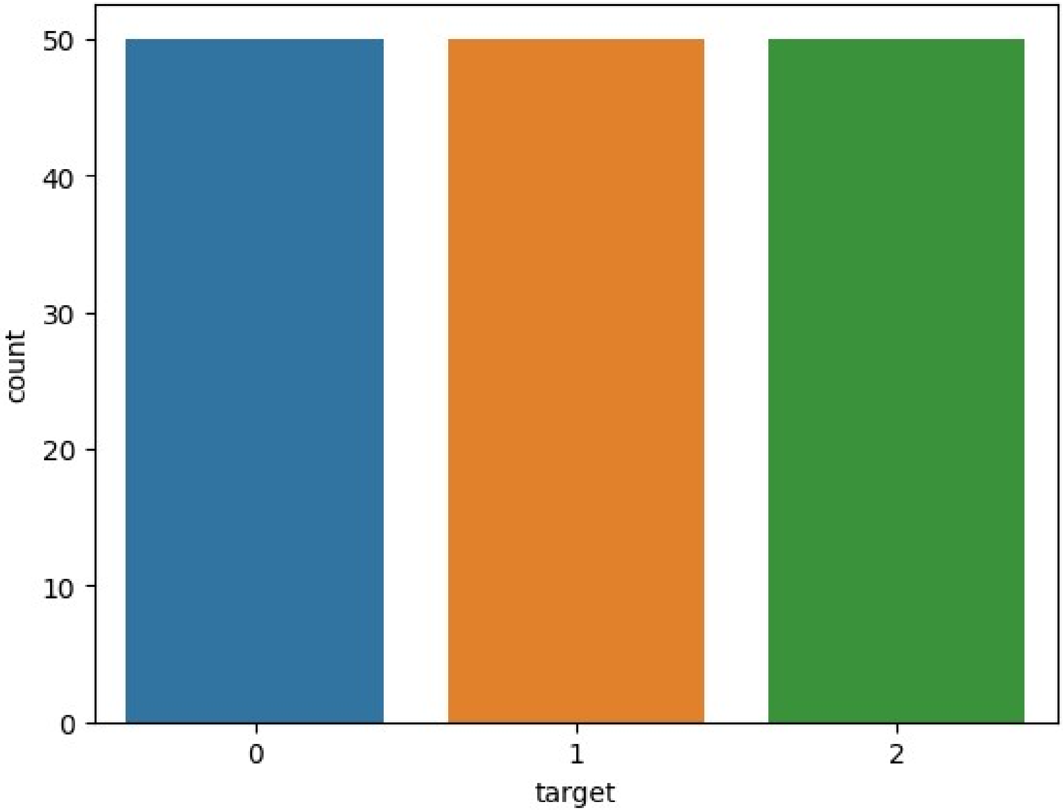
```
sns.distplot(df['sepal width (cm)'])
```

```
<AxesSubplot:xlabel='sepal width (cm)', ylabel='Density'>
```



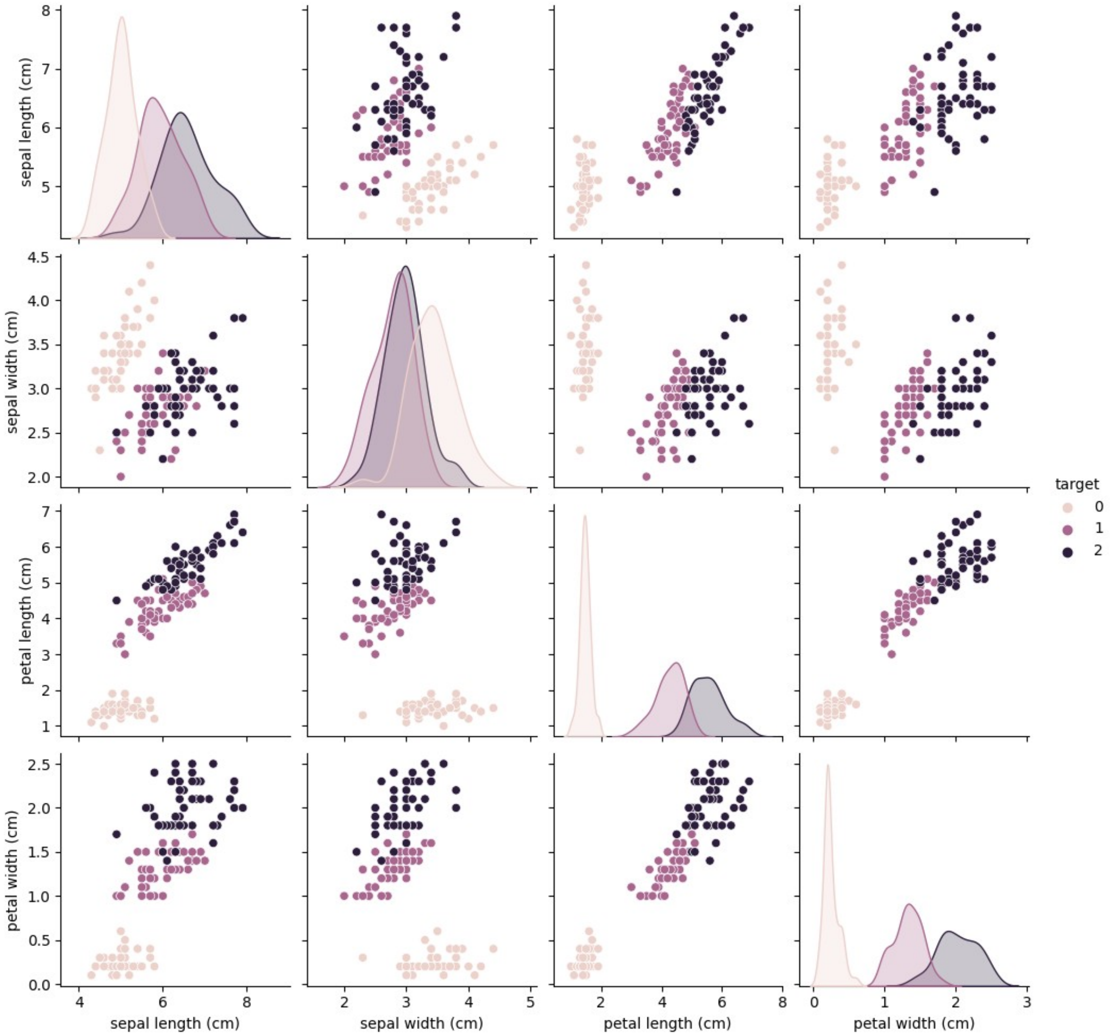
```
sns.countplot(df['target'])
```

```
<AxesSubplot:xlabel='target', ylabel='count'>
```



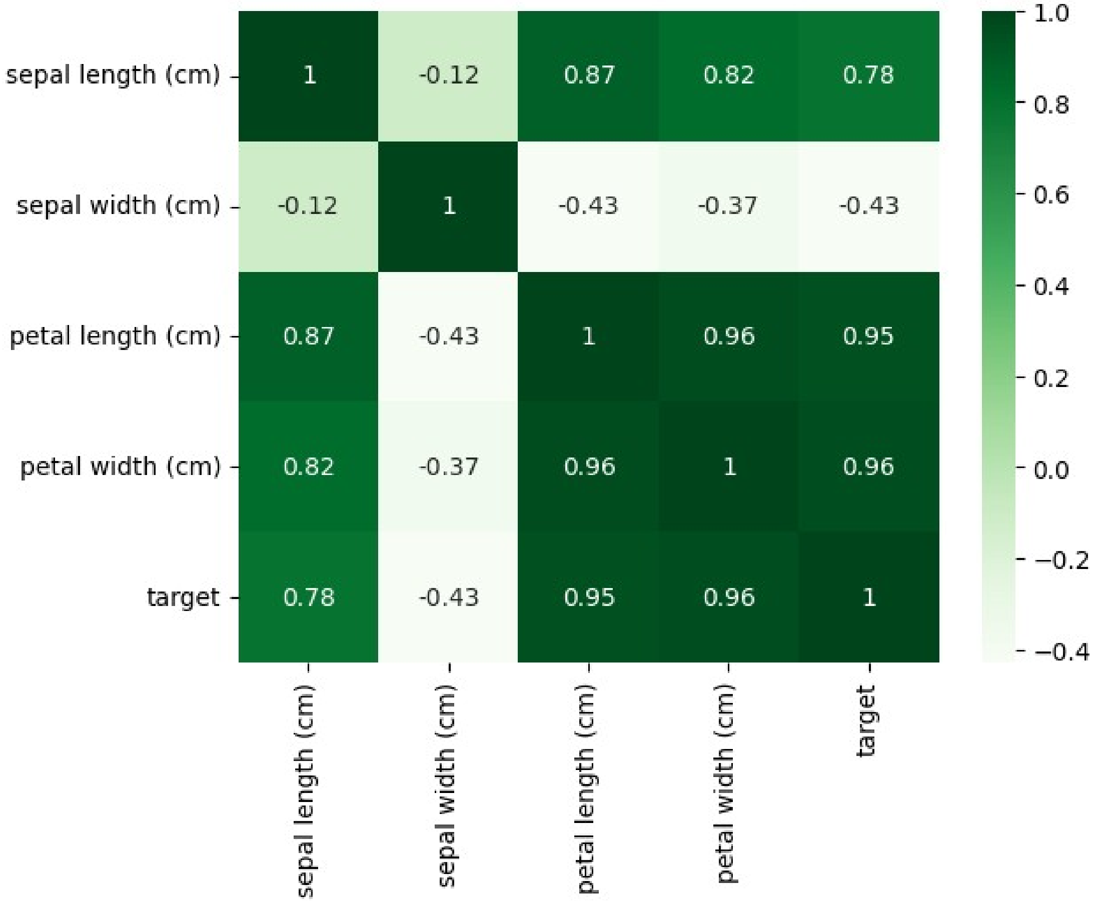
```
sns.pairplot(df,hue='target')
```

```
<seaborn.axisgrid.PairGrid at 0x1f69471f880>
```



```
sns.heatmap(df.corr(),annot=True,cmap='Greens')
```

```
<AxesSubplot:>
```



```
X=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.20,rand
om_state=123)
```

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.preprocessing import StandardScaler
```

```
X_test.shape
```

```
(30, 4)
```

```
X_train.shape
```

```
(120, 4)
```

```
y_test.shape
```

```
(30,)
```

```
y_train.shape
```

```
(120,)
```

```
sc=StandardScaler()
```

```
X_train=sc.fit_transform(X_train)
```

```
X_test=sc.transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
```

```
mlr=LogisticRegression()
```

```
model=OneVsRestClassifier(mlr)
```

```
model.fit(X_train,y_train)
```

```
OneVsRestClassifier(estimator=LogisticRegression())
```

```
y_train_pred=model.predict(X_train)
```

```
y_test_pred=model.predict(X_test)
```

```
from sklearn.metrics import classification_report
```

```
print('Train data')
classification_report(y_train,y_train_pred)
```

```
Train data
```

```
'              precision    recall  f1-score   support\n\n           0
1.00      1.00      1.00        37\n           1       0.93      0.93
0.93        44\n           2       0.92      0.92      0.92        39\
n\n    accuracy                           0.95       120\n   macro avg
0.95      0.95      0.95       120\nweighted avg       0.95      0.95
0.95       120\n'
```

```
print('Test data')
classification_report(y_test,y_test_pred)
```

```
Test data
```

```
'              precision    recall  f1-score   support\n\n           0
1.00      1.00      1.00        13\n           1       0.86      1.00
0.92         6\n           2       1.00      0.91      0.95        11\
n\n    accuracy                           0.97        30\n   macro avg
0.95      0.97      0.96        30\nweighted avg       0.97      0.97
0.97        30\n'
```

```
from sklearn.multiclass import OneVsOneClassifier
```

```
model1=OneVsOneClassifier(mlr)
```

```
model1.fit(X_train,y_train)
```

```
OneVsOneClassifier(estimator=LogisticRegression())
```

```
y_train_pred=model1.predict(X_train)
```

```
y_test_pred=model1.predict(X_test)
```

```
print('Train data')
print(classification_report(y_train,y_train_pred))
```

```
Train data
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        37
           1       0.98      0.95      0.97        44
           2       0.95      0.97      0.96        39

    accuracy                           0.97       120
   macro avg       0.98      0.98      0.98       120
weighted avg       0.98      0.97      0.98       120
```

```
print('Test data')
print(classification_report(y_test,y_test_pred))
```

```
Test data
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       0.86      1.00      0.92         6
           2       1.00      0.91      0.95        11

    accuracy                           0.97        30
   macro avg       0.95      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=3) #bydefault 5 neighbors
```

```
knn.fit(X_train,y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
from sklearn.metrics import accuracy_score
y_train_pred=knn.predict(X_train)
y_test_pred=knn.predict(X_test)
```

```
print("Train Data")
print(accuracy_score(y_train,y_train_pred))
print("Test Data")
print(accuracy_score(y_test,y_test_pred))
```

```
Train Data
0.9583333333333334
```

Test Data
0.9