Prediction using Unsupervised ML

Iris Dataset

From the given ‘Iris’ dataset, predict the optimum number of clusters and represent it visually.

Shital more

In [1]:
```
#Import Libraries
```

```
For this project I have used the following Libaries
```

In [3]:
```
import os  #Standard imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
```

In [9]:
```
from sklearn.datasets import load_iris
```

In [12]:
```
iris = load_iris()
```

Data Information ¶

```
Importing the csv file in dataframe
```

In [5]:
```
df=pd.read_csv("Iris.csv")
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
       'Species'],
      dtype='object')
```

In [13]:
```
iris.keys()
```

Out[13]:
```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filenam
e'])
```

In [19]:
```
iris.target_names
```

Out[19]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [15]:
```
df.head()
```

Out[15]:

	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

In [ ]:
```
##Mistake - Imported data using csv and also through sklearn.dataset
```

In [5]:
```
sns.set()
```

In [6]:
```
#datasets.load_iris?#documentation is going to pop up
```

In [7]:
```
data=datasets.load_iris()
```

In [8]:
```
data.keys()
```

Out[8]:
```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filenam
e'])
```

In [9]:
```
print(data["DESCR"])
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
```

```
    Structure and Classification Rule for Recognition in Partially Exposed
    Environments".  IEEE Transactions on Pattern Analysis and Machine
    Intelligence, Vol. PAMI-2, No. 1, 67-71.
  - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
    on Information Theory, May 1972, 431-433.
  - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
    conceptual clustering system finds 3 classes in the data.
  - Many, many more ...
```

In [10]:
```python
data["data"]
```

Out[10]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
```

In [11]:
```python
data["data"][:5]
```

Out[11]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

In [12]:
```python
data["feature_names"]
```

Out[12]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

In [13]:
```python
data["target"]
```

Out[13]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [14]:
```
data["target_names"]
```

Out[14]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

Create a Pandas DataFrame from the Data

In [22]:
```
pd.DataFrame?
```

In [23]:
```
pd.DataFrame(data["data"],columns=data["feature_names"])
```

Out[23]:

	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
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1
10	5.4	3.7	1.5	0.2
11	4.8	3.4	1.6	0.2
12	4.8	3.0	1.4	0.1
13	4.3	3.0	1.1	0.1
14	5.8	4.0	1.2	0.2
15	5.7	4.4	1.5	0.4
16	5.4	3.9	1.3	0.4
17	5.1	3.5	1.4	0.3
18	5.7	3.8	1.7	0.3
19	5.1	3.8	1.5	0.3
20	5.4	3.4	1.7	0.2
21	5.1	3.7	1.5	0.4
22	4.6	3.6	1.0	0.2
23	5.1	3.3	1.7	0.5
24	4.8	3.4	1.9	0.2
25	5.0	3.0	1.6	0.2
26	5.0	3.4	1.6	0.4
27	5.2	3.5	1.5	0.2
28	5.2	3.4	1.4	0.2
29	4.7	3.2	1.6	0.2
...	...	...	...	...
120	6.9	3.2	5.7	2.3
121	5.6	2.8	4.9	2.0
122	7.7	2.8	6.7	2.0
123	6.3	2.7	4.9	1.8

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
124	6.7	3.3	5.7	2.1
125	7.2	3.2	6.0	1.8
126	6.2	2.8	4.8	1.8
127	6.1	3.0	4.9	1.8
128	6.4	2.8	5.6	2.1
129	7.2	3.0	5.8	1.6
130	7.4	2.8	6.1	1.9
131	7.9	3.8	6.4	2.0
132	6.4	2.8	5.6	2.2
133	6.3	2.8	5.1	1.5
134	6.1	2.6	5.6	1.4
135	7.7	3.0	6.1	2.3
136	6.3	3.4	5.6	2.4
137	6.4	3.1	5.5	1.8
138	6.0	3.0	4.8	1.8
139	6.9	3.1	5.4	2.1
140	6.7	3.1	5.6	2.4
141	6.9	3.1	5.1	2.3
142	5.8	2.7	5.1	1.9
143	6.8	3.2	5.9	2.3
144	6.7	3.3	5.7	2.5
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 × 4 columns

In [24]:
```
df=pd.DataFrame(data["data"],columns=data["feature_names"])
```

In [25]:
```
df["target"]=data["target"]
```

In [26]:
```
df.head()
```

Out[26]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (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

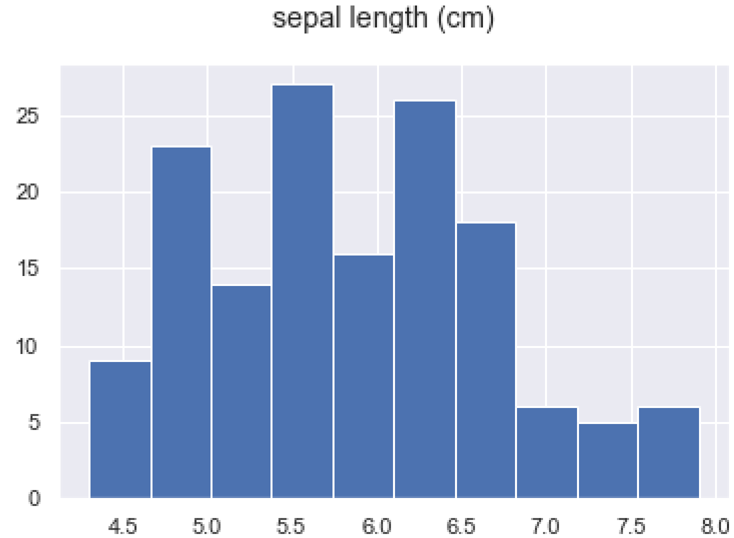
In [27]:
```
df.describe()    #To display stats about data
```

Out[27]:

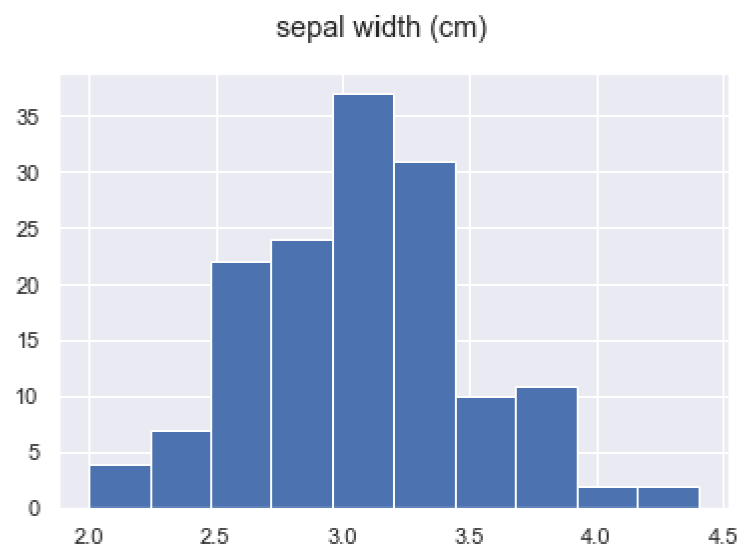
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	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

Exploratory Data Analysis

In [32]:
```
col="sepal length (cm)"
df[col].hist()
plt.suptitle(col)
```
Out[32]:
```
Text(0.5, 0.98, 'sepal length (cm)')
```

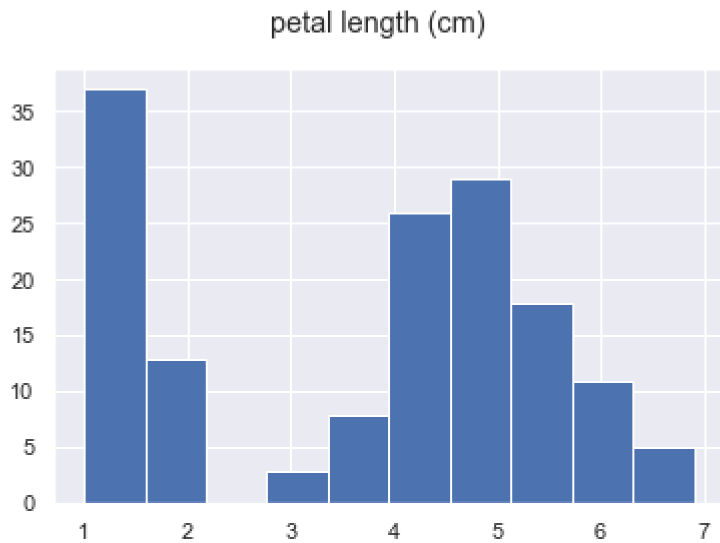


In [33]:
```
col="sepal width (cm)"
df[col].hist()
plt.suptitle(col)
```
Out[33]:
```
Text(0.5, 0.98, 'sepal width (cm)')
```



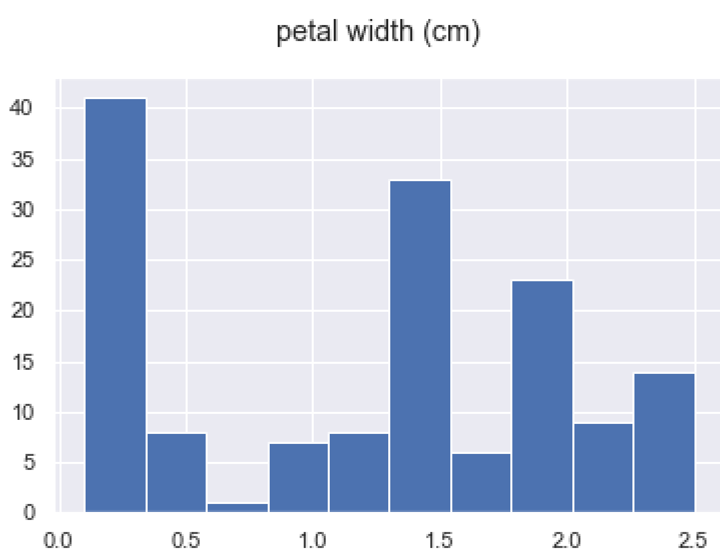
In [34]:
```
col="petal length (cm)"
df[col].hist()
plt.suptitle(col)
```

Out[34]: Text(0.5, 0.98, 'petal length (cm)')



In [35]:
```
col="petal width (cm)"
df[col].hist()
plt.suptitle(col)
```

Out[35]: Text(0.5, 0.98, 'petal width (cm)')



In [36]:
```
df["target"]
```

Out[36]:
```
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8      0
9      0
10     0
11     0
12     0
13     0
14     0
15     0
16     0
17     0
18     0
19     0
20     0
21     0
22     0
23     0
24     0
25     0
26     0
27     0
28     0
29     0
      ..
120    2
121    2
122    2
123    2
124    2
125    2
126    2
127    2
128    2
129    2
130    2
131    2
132    2
133    2
134    2
135    2
136    2
137    2
138    2
139    2
140    2
141    2
142    2
143    2
```

```
144    2
145    2
146    2
147    2
148    2
149    2
Name: target, Length: 150, dtype: int32
```

In [37]:
```
data["target_names"]
```

Out[37]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

In [38]:
```
df["target"].map({0:"setosa",1:"versicolor",2:"virginica"})
```

Out[38]:
```
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
5         setosa
6         setosa
7         setosa
8         setosa
9         setosa
10        setosa
11        setosa
12        setosa
13        setosa
14        setosa
15        setosa
16        setosa
17        setosa
18        setosa
19        setosa
20        setosa
21        setosa
22        setosa
23        setosa
24        setosa
25        setosa
26        setosa
27        setosa
28        setosa
29        setosa
         ...
120    virginica
121    virginica
122    virginica
123    virginica
124    virginica
125    virginica
126    virginica
127    virginica
128    virginica
129    virginica
130    virginica
131    virginica
132    virginica
133    virginica
134    virginica
135    virginica
136    virginica
137    virginica
138    virginica
139    virginica
140    virginica
141    virginica
142    virginica
143    virginica
```

```
144    virginica
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: target, Length: 150, dtype: object
```

In [39]:
```
df
```

Out[39]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (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
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5.0	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0
10	5.4	3.7	1.5	0.2	0
11	4.8	3.4	1.6	0.2	0
12	4.8	3.0	1.4	0.1	0
13	4.3	3.0	1.1	0.1	0
14	5.8	4.0	1.2	0.2	0
15	5.7	4.4	1.5	0.4	0
16	5.4	3.9	1.3	0.4	0
17	5.1	3.5	1.4	0.3	0
18	5.7	3.8	1.7	0.3	0
19	5.1	3.8	1.5	0.3	0
20	5.4	3.4	1.7	0.2	0
21	5.1	3.7	1.5	0.4	0
22	4.6	3.6	1.0	0.2	0
23	5.1	3.3	1.7	0.5	0
24	4.8	3.4	1.9	0.2	0
25	5.0	3.0	1.6	0.2	0
26	5.0	3.4	1.6	0.4	0
27	5.2	3.5	1.5	0.2	0
28	5.2	3.4	1.4	0.2	0
29	4.7	3.2	1.6	0.2	0
...	...	...	...	...	...
120	6.9	3.2	5.7	2.3	2
121	5.6	2.8	4.9	2.0	2
122	7.7	2.8	6.7	2.0	2
123	6.3	2.7	4.9	1.8	2

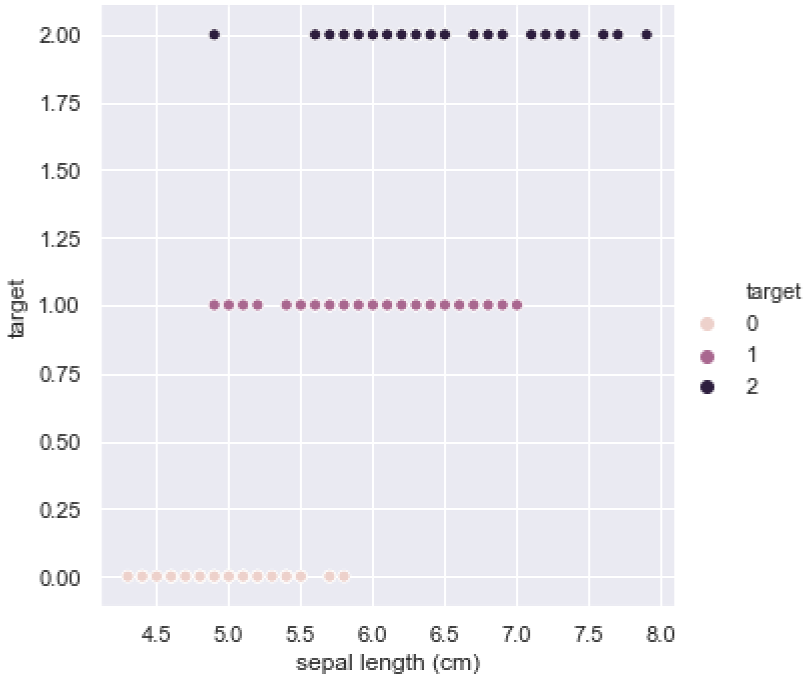
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
124	6.7	3.3	5.7	2.1	2
125	7.2	3.2	6.0	1.8	2
126	6.2	2.8	4.8	1.8	2
127	6.1	3.0	4.9	1.8	2
128	6.4	2.8	5.6	2.1	2
129	7.2	3.0	5.8	1.6	2
130	7.4	2.8	6.1	1.9	2
131	7.9	3.8	6.4	2.0	2
132	6.4	2.8	5.6	2.2	2
133	6.3	2.8	5.1	1.5	2
134	6.1	2.6	5.6	1.4	2
135	7.7	3.0	6.1	2.3	2
136	6.3	3.4	5.6	2.4	2
137	6.4	3.1	5.5	1.8	2
138	6.0	3.0	4.8	1.8	2
139	6.9	3.1	5.4	2.1	2
140	6.7	3.1	5.6	2.4	2
141	6.9	3.1	5.1	2.3	2
142	5.8	2.7	5.1	1.9	2
143	6.8	3.2	5.9	2.3	2
144	6.7	3.3	5.7	2.5	2
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

In [40]:
```
sns.relplot?
```

In [41]:
```
col="sepal length (cm)"
sns.relplot(x=col, y="target", hue="target",data=df)
```

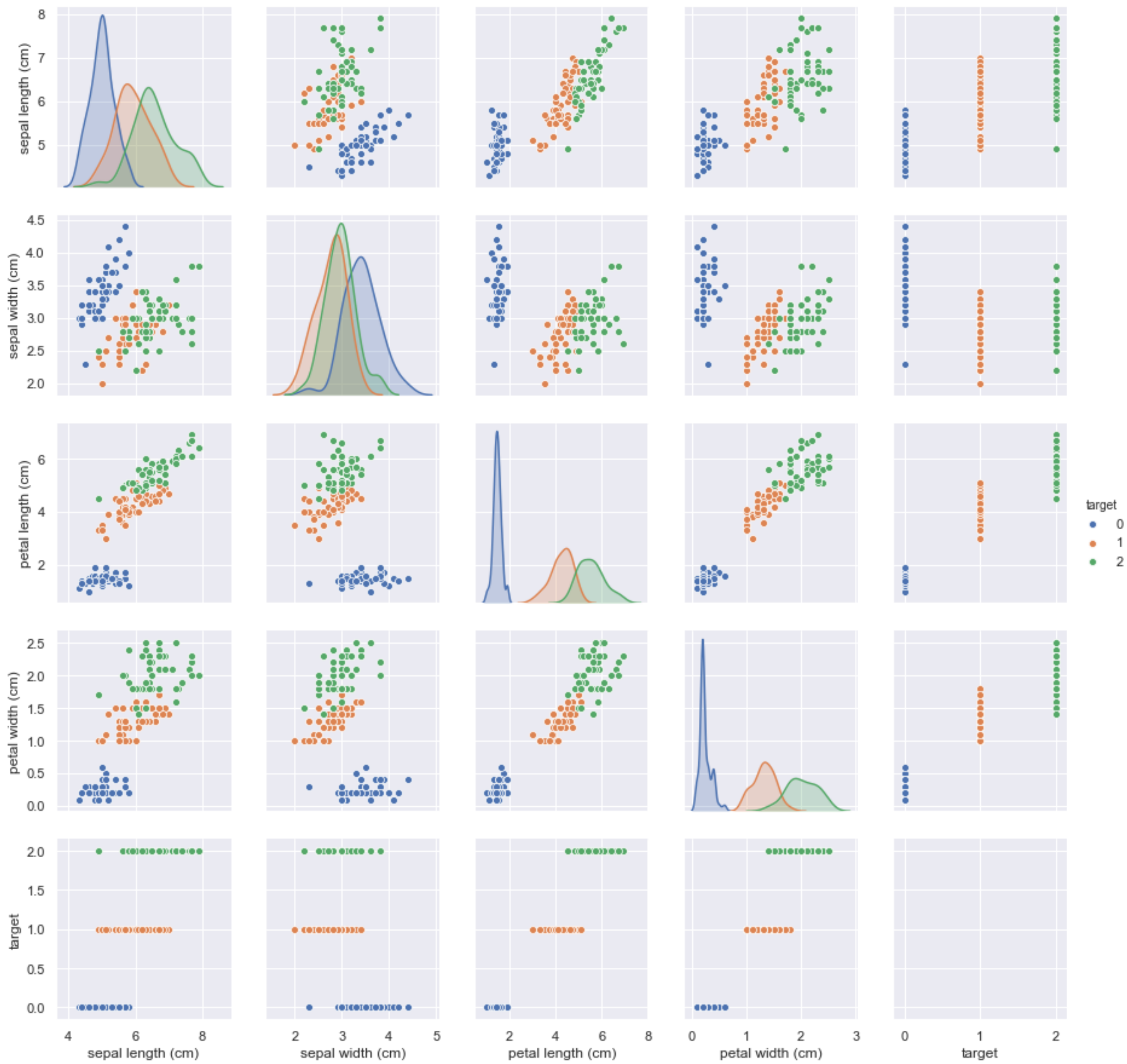
Out[41]: <seaborn.axisgrid.FacetGrid at 0x1db5f8a7400>



In [70]:
```
sns.pairplot(df, hue="target")
```

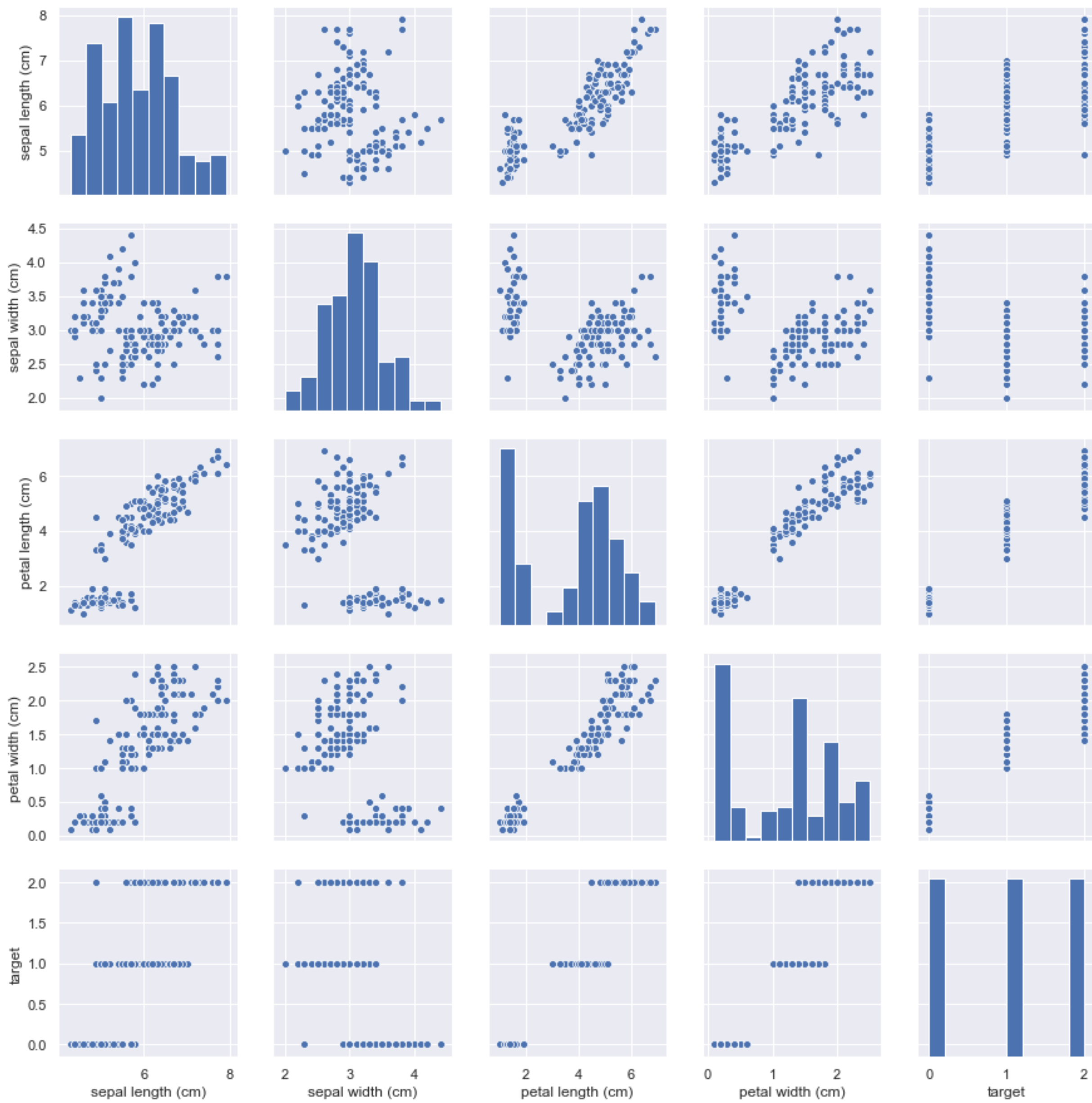
```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:48
7: RuntimeWarning: invalid value encountered in true_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p
y:34: RuntimeWarning: invalid value encountered in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
```

Out[70]: <seaborn.axisgrid.PairGrid at 0x199f62e12e8>



In [42]:
```
sns.pairplot(df)
```

Out[42]: <seaborn.axisgrid.PairGrid at 0x1db5f931c50>



In [43]:
```
df.head()
```

Out[43]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (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

Model training

train=70 test=30

In [44]:
```
from sklearn.model_selection import train_test_split
x=df.drop(columns=['target'])
y=df['target']
x_train, x_test, y_train , y_test=train_test_split(x,y,test_size=0.30)
```

LogisticRegression

In [45]:
```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

In [46]:
```
model.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:46
9: FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specif
y the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

Out[46]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [47]:
```
print("Accuracy:",model.score(x_test,y_test)*100)  #print metric to get performa
```

```
Accuracy: 93.33333333333333
```

KNN=K-nearest neighbors

In [48]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=1)
```

In [49]:
```
model.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:43
2: FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:46
9: FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specif
y the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

Out[49]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [116]:
```
print("Accuracy:",model.score(x_test,y_test)*100)
```

```
Accuracy: 97.77777777777777
```

Conclusion::- We are trying to use attributes of flowers to predict the species of the flower. specifically we are trying to use the sepal lenght and width ,the peatl lenght and width to predict if an iris flower is of type setosa ,versicolor or virginica. This is a multiclass classification problem.

In [ ]:
```

```