In [1]:
```
#importing basic libararies
import numpy as np
import pandas as pd
from sklearn import datasets
```

In [2]:
```
#importing iris dataset from sklearn dataset
iris_data = datasets.load_iris()
```

In [3]:
```
# lets have a look at the dataset
iris_data
```

```
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
```

In [4]:
```
#checking the type of dataset
type(iris_data)
```

Out[4]:
```
sklearn.utils.Bunch
```

In [5]:
```
#looking at the keys of the sklearn iris dataset
dir(iris_data)
```

Out[5]:
```
['DESCR', 'data', 'feature_names', 'filename', 'target', 'target_names']
```

In [6]:
```
#converting iris dataset into iris data frame called 'iris_df'
iris_df = pd.DataFrame(data = np.c_[iris_data['data'],iris_data['target']], colu
# I looked at this source : https://stackoverflow.com/questions/38105539/how-to-
```

Doing basic Exploratory data analysis on the given data

In [7]:
```
#seeing how data looks like
iris_df.head()
```

Out[7]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

So, we have features as 'sepal length (cm)', 'sepal width (cm)', 'petal length(cm)' and 'petal width(cm)'.
The output is either '0.0', '1.0' or '2.0' which is stored in 'target' column of our dataset.
Form the printed 'iris_data' above we can see : target '0.0' corresponds to class 'setosa', target '1.0' corresponds to class 'versicolor'and target '2.0' corresponds to class 'virginica'.

In [8]:
```
# describing the data
iris_df.describe().T
```

Out[8]:

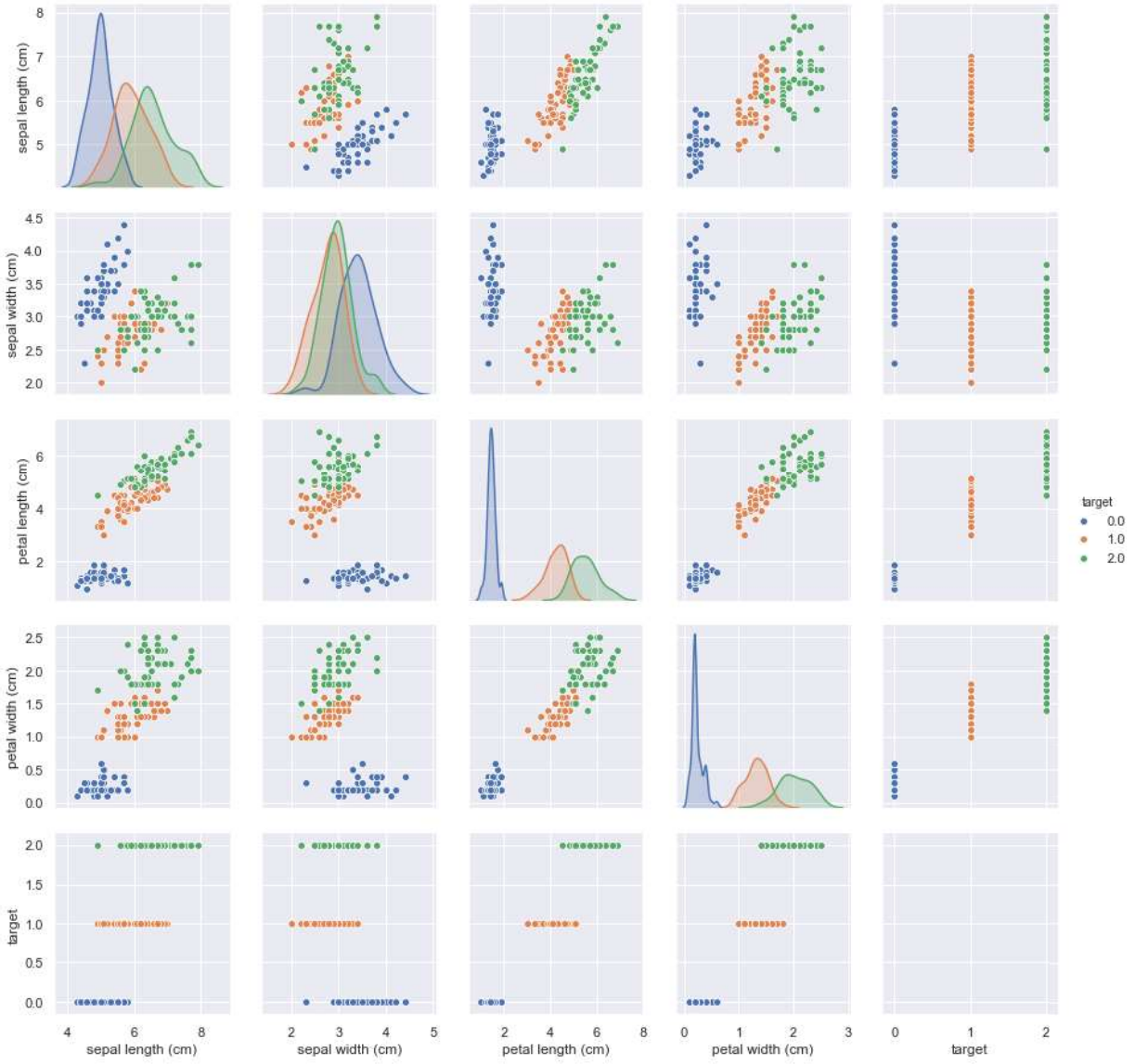
	count	mean	std	min	25%	50%	75%	max
sepal length (cm)	150.0	5.843333	0.828066	4.3	5.1	5.80	6.4	7.9
sepal width (cm)	150.0	3.057333	0.435866	2.0	2.8	3.00	3.3	4.4
petal length (cm)	150.0	3.758000	1.765298	1.0	1.6	4.35	5.1	6.9
petal width (cm)	150.0	1.199333	0.762238	0.1	0.3	1.30	1.8	2.5
target	150.0	1.000000	0.819232	0.0	0.0	1.00	2.0	2.0

In [9]:
```
# let us understand the dataframe information
#, lets see if there is any missing data and understand datatype
iris_df.info(verbose =True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal length (cm)    150 non-null float64
sepal width (cm)     150 non-null float64
petal length (cm)    150 non-null float64
petal width (cm)     150 non-null float64
target               150 non-null float64
dtypes: float64(5)
memory usage: 5.9 KB
```

In [10]:
```
#importing libraries for plotting the graphs
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [11]:
```
#plotting scatter plot among all the features and targets
#p = sns.pairplot(iris_df)
p = sns.pairplot(iris_df, hue = "target")
#g = sns.pairplot(iris_df, hue = "target", vars = iris_df.columns[:-1] )
```



The above graphs statistics tell us :

1. target 0.0 i.e. 'setosa' flower has smallest value for mean sepal length, mean petal length and mean petal width also it has largest mean for sepal width compared to other targets in the dataset.

2. target 1.0 i.e. 'versicolor' flower has avergae value for mean sepal length, mean petal length and petal width compared to other targets in the dataset and smallest sepal width among all targets.
3. target 2.0 i.e. 'virginica' flower has largest value for mean sepal length, mean petal length and mean petal width.

In [12]:
```
#making a copy of dataset to train the model
iris_df2 = iris_df.copy(deep =True)
```

In [13]:
```
#importing useful libraries
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [14]:
```
#preprocessing data
X = iris_df2.drop(['target'], axis =1)
y = iris_df2.target
print ("shape of X is :", X.shape)
print("shape of y is :", y.shape)
```

```
shape of X is : (150, 4)
shape of y is : (150,)
```

KNN Classifier

In [15]:
```
#splitting data into training, validation and test sets
X_train_k, X_test_k, y_train_k, y_test_k = train_test_split(X, y, test_size = 0.
X_train_k, X_valid_k, y_train_k, y_valid_k = train_test_split(X_train_k, y_train_
print( "X_train_k.shape is :", X_train_k.shape)
print( "X_valid_k.shape is :", X_valid_k.shape)
print( "X_valid_k.shape is :", X_valid_k.shape)
print( "y_train_k.shape is :", y_train_k.shape)
print( "y_valid_k.shape is :", y_valid_k.shape)
print( "y_valid_k.shape is :", y_valid_k.shape)
```

```
X_train_k.shape is : (90, 4)
X_valid_k.shape is : (30, 4)
X_valid_k.shape is : (30, 4)
y_train_k.shape is : (90,)
y_valid_k.shape is : (30,)
y_valid_k.shape is : (30,)
```

In [16]:
```
# lets have a look at X_trian_k X_test_k and X_valid_k to make sure that they ar
print("\n")
print("Description of training set feature values ")
print(X_train_k.describe().T)
print("\n")
print("Description of validation set feature values ")
print(X_valid_k.describe().T)
print("\n")
print("Description of testing set feature values ")
print(X_test_k.describe().T)
```

```
Description of training set feature values
                   count      mean       std  min  25%  50%  75%  max
sepal length (cm)   90.0  5.846667  0.835074  4.3  5.1  5.8  6.4  7.7
sepal width (cm)    90.0  3.112222  0.462000  2.0  2.8  3.0  3.4  4.4
petal length (cm)   90.0  3.727778  1.797753  1.1  1.5  4.3  5.1  6.7
petal width (cm)    90.0  1.188889  0.769568  0.1  0.3  1.3  1.8  2.5


Description of validation set feature values
                   count      mean       std  min    25%   50%    75%  max
sepal length (cm)   30.0  5.696667  0.791978  4.4  5.200  5.70  6.225  7.7
sepal width (cm)    30.0  2.910000  0.375408  2.2  2.625  3.00  3.100  3.6
petal length (cm)   30.0  3.723333  1.637636  1.0  1.750  4.10  4.975  6.1
petal width (cm)    30.0  1.166667  0.710189  0.2  0.225  1.35  1.750  2.3


Description of testing set feature values
                   count      mean       std  min    25%   50%    75%  max
sepal length (cm)   30.0  5.980000  0.845026  4.7  5.425  6.05  6.500  7.9
sepal width (cm)    30.0  3.040000  0.384708  2.2  2.800  3.00  3.200  3.8
petal length (cm)   30.0  3.883333  1.841305  1.3  1.600  4.50  5.175  6.9
petal width (cm)    30.0  1.263333  0.810910  0.1  0.325  1.35  2.000  2.3
```

In [17]:
```
#Lets see performance of KNN classifier, default case on this dataset
Knn_default = KNeighborsClassifier()
Knn_default.fit(X_train_k, y_train_k)
y_pred_test = Knn_default.predict(X_test_k)
score_default = accuracy_score(y_test_k, y_pred_test)
print(" accuracy on default Knn is : ", score_default , sep ="\t")
```

```
 accuracy on default Knn is :   0.9666666666666667
```

The KNN on default case gives an approximate accuracy of 0.967 on the test set