Objective:

The aim is to classify iris flowers among three species (setose, versicolor or virginica) from measurements of length and width of sepals and petals. The iris data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant.

The central goal here is to design a model which makes good classifications for a new flower or, in other words, one which exhibits good generalization.

In [2]:

```
#Standard libaries

import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
```

About the data :

The data we use for this example is the iris dataset ,a classical dataset in machine learning and statistics. It is included in scikit-learn in the dataset module. We can load it by calling the load_iris function.

In [3]:

```
from sklearn.datasets import load_iris
iris=load_iris()
```

The iris object that is returned by load_iris is a bunch object,which is very similar to a dictionary .It contains keys and values

In [4]:

```
iris.keys()
```

Out[4]:

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'file
name'])
```

Description of Data set IRIS

> ['DESCR'] is the description of the Data set .It involues the Number of Instances,Number of Attributes,summary statistics and little more about the Data usages .

In [5]:

```
print(iris['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 take
n
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 an
d
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 Analys
is.
```

```
      (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 Transacti
ons
      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 ...
```

The value with key target_names is an array of strings,containing the species of flower that we want to predict......

In [6]:

```
iris['target_names']
```

Out[6]:

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

The feature_names are a list of strings,giving the description of each features

In [7]:

```
iris['feature_names']
```

Out[7]:

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

In [8]:

```
type(iris['data'])
```

Out[8]:

```
numpy.ndarray
```

In [9]:

```
iris['data'].shape
```

Out[9]:

```
(150, 4)
```

Here are the feature values for the first five samples :

In [10]:

```
iris['data'][:5]
```

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]])
```

The species are Encoded as integers from 0 to 2..

In [11]:

```
iris['target']
```

Out[11]:

```
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, 2, 2])
```

The meaning of the numbers are given by the iris['target_names'] array :0 means setosa,1 means versicolor and 2 means Virginica.

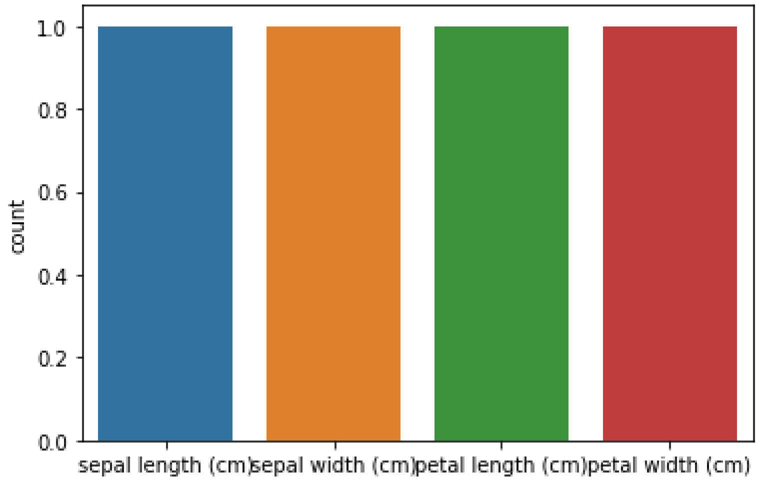
Data visualization

In [12]:

```
sns.countplot(x="feature_names",data=iris)
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d5b40e2808>
```

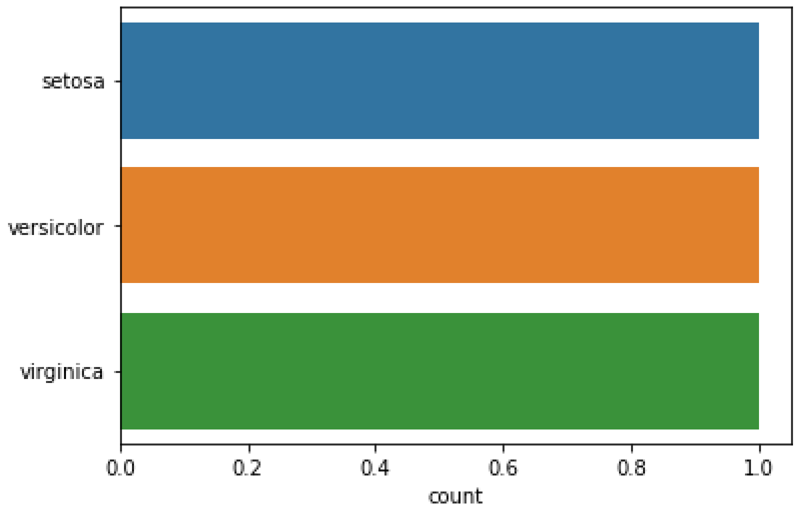


In [13]:

```
sns.countplot(y="target_names",data=iris)
```

Out[13]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d5b59b4d88>
```



Measuring success:Training and Testing data

The part of the data is used to build our machine learning model,and is calling the training data ot training set.The rest of the data will be used well the model works and is called test data,test set or hold-out set.

In [14]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(iris['data'],iris['target'],random_state=0
```

The output of the train_test_split function are x_train,x_test,y_train,y_test which are all numpy arrays.x_train contains 75% of the rows of the dataset,and x_test contains the remaining 25%

In [15]:

```
x_train.shape
```

Out[15]:

```
(112, 4)
```

In [16]:

```
x_test.shape
```

Out[16]:

```
(38, 4)
```

Nearest K Neighbor:

In [17]:

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=1)
```

In [18]:

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

Out[18]:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

In [19]:

```
import numpy as np
x_new=np.array([[5,2.9,1,0.4]])
x_new.shape
```

Out[19]:

```
(1, 4)
```

To make prediction we call the predict method of the knn object:

In [20]:

```
prediction=knn.predict(x_new)
prediction
```

Out[20]:

```
array([0])
```

In [21]:

```
iris['target_names'][prediction]
```

Out[21]:

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

Evaluating the Model

In [22]:

```
knn.score(x_test,y_test)
```

Out[22]:

```
0.9736842105263158
```

In [ ]:

```

```