Question:2 (Solution)

```
import numpy as np
from sklearn import datasets

iris = datasets.load_iris()

print(iris.DESCR)
```

```
**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 ...
```

```
x_iris = iris.data
y_iris = iris.target

print("The shape of the NumPy array x_iris containing the predictors is:", x_iris.shape)
print("The shape of the NumPy array y_iris containing the label is:",y_iris.shape)
print (y_iris)
```

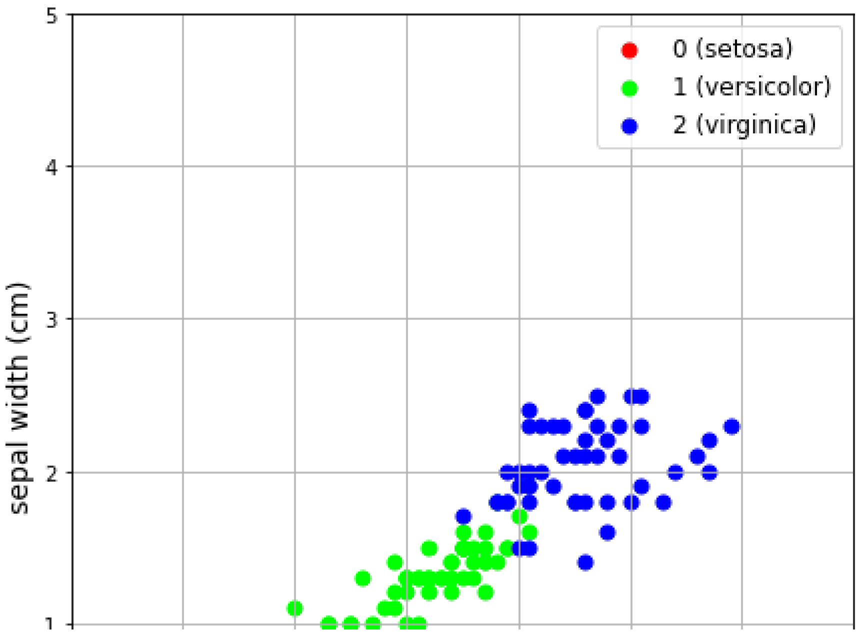
```
The shape of the NumPy array x_iris containing the predictors is: (150, 4)
The shape of the NumPy array y_iris containing the label is: (150,)
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 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]
```

```
x_iris_petal = iris.data[:, 2:]
print (x_iris_petal.shape)
print (x_iris_petal)
print("The last two predictors are", iris.feature_names[2:])
print("The shape of the NumPy array x_iris_petal containing the last two predictors is", x
```

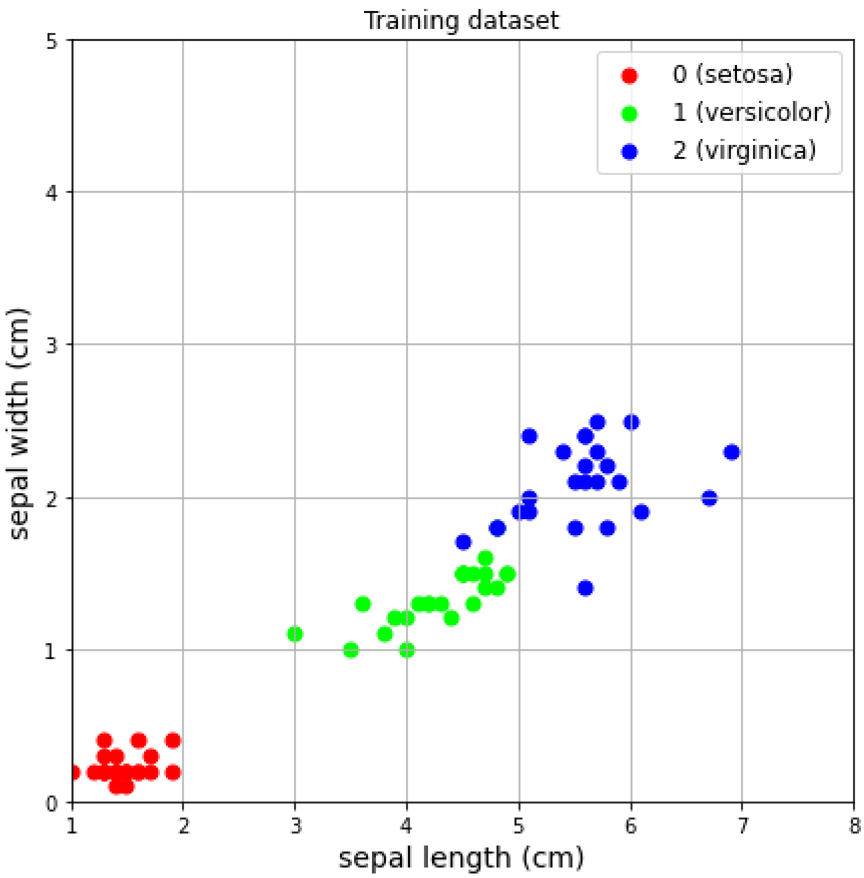
```
 [4.2 1.2]
 [4.2 1.3]
 [4.3 1.3]
 [3.  1.1]
 [4.1 1.3]
 [6.  2.5]
 [5.1 1.9]
 [5.9 2.1]
 [5.6 1.8]
 [5.8 2.2]
 [6.6 2.1]
 [4.5 1.7]
 [6.3 1.8]
 [5.8 1.8]
 [6.1 2.5]
 [5.1 2. ]
 [5.3 1.9]
 [5.5 2.1]
 [5.  2. ]
 [5.1 2.4]
 [5.3 2.3]
 [5.5 1.8]
 [6.7 2.2]
 [6.9 2.3]
 [5.  1.5]
```

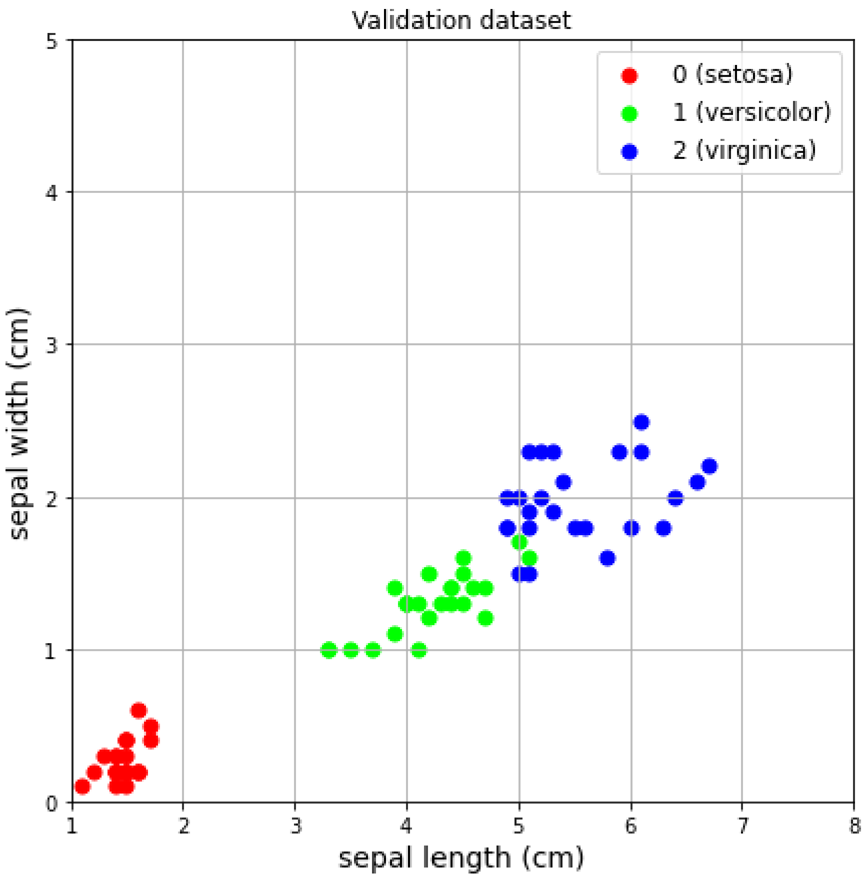
```
 [5.7 2.3]
 [4.9 2. ]
 [6.7 2. ]
 [4.9 1.8]
 [5.7 2.1]
 [6.  1.8]
 [4.8 1.8]
 [4.9 1.8]
 [5.6 2.1]
 [5.8 1.6]
 [6.1 1.9]
 [6.4 2. ]
 [5.6 2.2]
 [5.1 1.5]
 [5.6 1.4]
 [6.1 2.3]
 [5.6 2.4]
 [5.5 1.8]
 [4.8 1.8]
 [5.4 2.1]
 [5.6 2.4]
 [5.1 2.3]
 [5.1 1.9]
 [5.9 2.3]
 [5.7 2.5]
 [5.2 2.3]
 [5.  1.9]
 [5.2 2. ]
 [5.4 2.3]
 [5.1 1.8]]
The last two predictors are ['petal length (cm)', 'petal width (cm)']
The shape of the NumPy array x_iris_petal containing the last two predictors is (1
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(7, 7))

scatter = plt.scatter(x_iris_petal[y_iris==0,0], x_iris_petal[y_iris==0,1], s=50 , color=
scatter = plt.scatter(x_iris_petal[y_iris==1,0], x_iris_petal[y_iris==1,1], s=50 , color=
scatter = plt.scatter(x_iris_petal[y_iris==2,0], x_iris_petal[y_iris==2,1], s=50 , color=

plt.legend(fontsize=12)
plt.xlabel(iris.feature_names[0], fontsize=14)
plt.ylabel(iris.feature_names[1], fontsize=14)
plt.xlim(1,8)
plt.ylim(0,5)
plt.grid(True)
plt.show()
```




```
# Here we split our dataset into training and validation datasets
x_train = x_iris_petal[::2,:] # train data set (both predictors)
y_train = y_iris[::2] # train data set (labels)
x_val = x_iris_petal[1::2,:] # test data set (both predictors)
y_val = y_iris[1::2] # test data set (labels)


# The rest of this cell is used to plot the training and validation datasets
plt.figure(figsize=(7, 7))

scatter = plt.scatter(x_train[y_train==0,0], x_train[y_train==0,1], s=50 , color= '#FF0000
scatter = plt.scatter(x_train[y_train==1,0], x_train[y_train==1,1], s=50 , color= '#00FF00
scatter = plt.scatter(x_train[y_train==2,0], x_train[y_train==2,1], s=50 , color= '#0000FF

plt.title("Training dataset")
plt.legend(fontsize=12)
plt.xlabel(iris.feature_names[0], fontsize=14)
plt.ylabel(iris.feature_names[1], fontsize=14)
plt.xlim(1,8)
plt.ylim(0,5)
plt.grid(True)

plt.figure(figsize=(7, 7))

scatter = plt.scatter(x_val[y_val==0,0], x_val[y_val==0,1], s=50 , color= '#FF0000', label
scatter = plt.scatter(x_val[y_val==1,0], x_val[y_val==1,1], s=50 , color= '#00FF00', label
scatter = plt.scatter(x_val[y_val==2,0], x_val[y_val==2,1], s=50 , color= '#0000FF', label

plt.title("Validation dataset")
plt.legend(fontsize=12)
plt.xlabel(iris.feature_names[0], fontsize=14)
plt.ylabel(iris.feature_names[1], fontsize=14)
plt.xlim(1,8)
plt.ylim(0,5)
plt.grid(True)
plt.show()
```






```
from google.colab import widgets
from sklearn import neighbors
#from matplotlib.colors import ListedColormap

k_values = range(1,75,6)
```

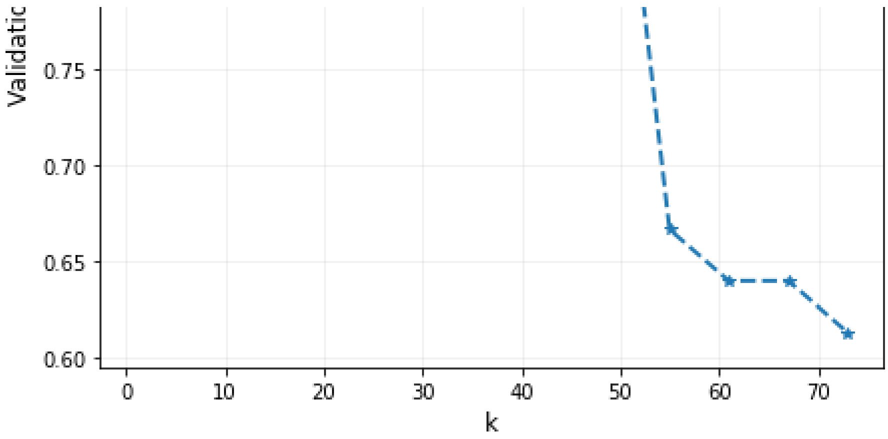
```
tb = widgets.TabBar([str(k) for k in k_values])

#cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])

accuracy = dict.fromkeys(k_values)

for k in k_values:
  with tb.output_to(str(k), select= (k < 2)):

    # First we create the kNN model
    knn = neighbors.KNeighborsClassifier(n_neighbors=k)
    knn.fit(x_train, y_train)



    # Finally we calculate the validation accuracy
    y_val_pred = knn.predict(x_val)
    accuracy[k] = np.sum(y_val==y_val_pred)/len(y_val)

    print("The validation accuracy for k=", k, "is ", accuracy[k])


# Here we predict the value of the validation accuracy as a function of k
plt.figure(figsize=(7, 7))
plt.plot(k_values, list(accuracy.values()), '--*', linewidth=2)
plt.xlabel("k", fontsize=12)
plt.ylabel("Validation accuracy", fontsize=12)
plt.grid(alpha=0.2)
plt.show()
```

1 7 13 19 25 31 37 **43** 49 55 61 67 73

The validation accuracy for k= 43 is 0.9733333333333334

```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=43)

# Train the model using the training sets
model.fit(x_iris_petal,y_iris)

#Predict Output
predicted= model.predict([[4,2]])
print(predicted)
```

```
[1]
```



✓ 0s completed at 00:47