Keras

In [1]:
```
import numpy as np
```

In [2]:
```
from sklearn.datasets import load_iris
```

In [3]:
```
iris = load_iris()
```

In [4]:
```
type(iris)
```

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

In [6]:
```
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 cl
ass
    :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 i
s taken
from Fisher's paper. Note that it's the same as in R, but not as in th
e 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 fi
eld 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 ref
ers 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 probl
ems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributio
ns 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 Sys
tem
     Structure and Classification Rule for Recognition in Partially Ex
posed
     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 Tra
nsactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLA
SS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

In [7]:
```
# X
X = iris.data
```

In [12]:
```
X
```

Out[12]:
```
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 [8]:
```
# y
y = iris.target
```

In [10]:
```
y
```

Out[10]:
```
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, 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])
```

Observation: y has labels and this is a numpy array.

We are now going to transform the labels.

We will use One-hot encoding.

class 0 --> [1, 0, 0] At index 0, we have 1 and 0 everywhere else.

class 1 --> [0, 1, 0] At index 1, we have 1 and 0 everywhere else.

class 2 --> [0, 0, 1] At index 2, we have 1 and 0 everywhere else.

In [13]:
```
from keras.utils import to_categorical
```

In [14]:
```
y = to_categorical(y)
```

In [15]:
```
y.shape
```

Out[15]:
```
(150, 3)
```

In [17]:
```
y[0: 10]   # First 10 rows of y
```

Out[17]:
```
array([[1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.]], dtype=float32)
```

In [18]:
```
y [-10: ]  # Last 10 rows of y
```

Out[18]:
```
array([[0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.]], dtype=float32)
```

In [19]:
```
from sklearn.model_selection import train_test_split
```

In [20]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33
                                                    random_state=42)
```

In [22]:
```
X_train
```

Out[22]:
```
array([[5.7, 2.9, 4.2, 1.3],
       [7.6, 3. , 6.6, 2.1],
       [5.6, 3. , 4.5, 1.5],
       [5.1, 3.5, 1.4, 0.2],
       [7.7, 2.8, 6.7, 2. ],
       [5.8, 2.7, 4.1, 1. ],
       [5.2, 3.4, 1.4, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [5.1, 3.8, 1.9, 0.4],
       [5. , 2. , 3.5, 1. ],
       [6.3, 2.7, 4.9, 1.8],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [5.6, 2.7, 4.2, 1.3],
       [5.1, 3.4, 1.5, 0.2],
       [5.7, 3. , 4.2, 1.2],
       [7.7, 3.8, 6.7, 2.2],
       [4.6, 3.2, 1.4, 0.2],
       [6.2, 2.9, 4.3, 1.3],
       [5.7, 2.5, 5. , 2. ],
       [5.5, 4.2, 1.4, 0.2],
       [6. , 3. , 4.8, 1.8],
       [5.8, 2.7, 5.1, 1.9],
       [6. , 2.2, 4. , 1. ],
       [5.4, 3. , 4.5, 1.5],
       [6.2, 3.4, 5.4, 2.3],
       [5.5, 2.3, 4. , 1.3],
       [5.4, 3.9, 1.7, 0.4],
       [5. , 2.3, 3.3, 1. ],
       [6.4, 2.7, 5.3, 1.9],
       [5. , 3.3, 1.4, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 2.4, 3.8, 1.1],
       [6.7, 3. , 5. , 1.7],
       [4.9, 3.1, 1.5, 0.2],
       [5.8, 2.8, 5.1, 2.4],
       [5. , 3.4, 1.5, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.9, 3.2, 4.8, 1.8],
       [5.1, 2.5, 3. , 1.1],
       [6.9, 3.2, 5.7, 2.3],
       [6. , 2.7, 5.1, 1.6],
       [6.1, 2.6, 5.6, 1.4],
```

```
       [7.7, 3. , 6.1, 2.3],
       [5.5, 2.5, 4. , 1.3],
       [4.4, 2.9, 1.4, 0.2],
       [4.3, 3. , 1.1, 0.1],
       [6. , 2.2, 5. , 1.5],
       [7.2, 3.2, 6. , 1.8],
       [4.6, 3.1, 1.5, 0.2],
       [5.1, 3.5, 1.4, 0.3],
       [4.4, 3. , 1.3, 0.2],
       [6.3, 2.5, 4.9, 1.5],
       [6.3, 3.4, 5.6, 2.4],
       [4.6, 3.4, 1.4, 0.3],
       [6.8, 3. , 5.5, 2.1],
       [6.3, 3.3, 6. , 2.5],
       [4.7, 3.2, 1.3, 0.2],
       [6.1, 2.9, 4.7, 1.4],
       [6.5, 2.8, 4.6, 1.5],
       [6.2, 2.8, 4.8, 1.8],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 5.3, 2.3],
       [5.1, 3.8, 1.6, 0.2],
       [6.9, 3.1, 5.4, 2.1],
       [5.9, 3. , 4.2, 1.5],
       [6.5, 3. , 5.2, 2. ],
       [5.7, 2.6, 3.5, 1. ],
       [5.2, 2.7, 3.9, 1.4],
       [6.1, 3. , 4.6, 1.4],
       [4.5, 2.3, 1.3, 0.3],
       [6.6, 2.9, 4.6, 1.3],
       [5.5, 2.6, 4.4, 1.2],
       [5.3, 3.7, 1.5, 0.2],
       [5.6, 3. , 4.1, 1.3],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [5.1, 3.7, 1.5, 0.4],
       [4.9, 2.4, 3.3, 1. ],
       [6.7, 3.3, 5.7, 2.5],
       [7.2, 3. , 5.8, 1.6],
       [4.9, 3.6, 1.4, 0.1],
       [6.7, 3.1, 5.6, 2.4],
       [4.9, 3. , 1.4, 0.2],
       [6.9, 3.1, 4.9, 1.5],
       [7.4, 2.8, 6.1, 1.9],
       [6.3, 2.9, 5.6, 1.8],
       [5.7, 2.8, 4.1, 1.3],
       [6.5, 3. , 5.5, 1.8],
       [6.3, 2.3, 4.4, 1.3],
       [6.4, 2.9, 4.3, 1.3],
       [5.6, 2.8, 4.9, 2. ],
       [5.9, 3. , 5.1, 1.8],
```

```
       [5.4, 3.4, 1.7, 0.2],
       [6.1, 2.8, 4. , 1.3],
       [4.9, 2.5, 4.5, 1.7],
       [5.8, 4. , 1.2, 0.2],
       [5.8, 2.6, 4. , 1.2],
       [7.1, 3. , 5.9, 2.1]])
```

In [23]:
```
X_test
```

Out[23]:
```
array([[6.1, 2.8, 4.7, 1.2],
       [5.7, 3.8, 1.7, 0.3],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.9, 4.5, 1.5],
       [6.8, 2.8, 4.8, 1.4],
       [5.4, 3.4, 1.5, 0.4],
       [5.6, 2.9, 3.6, 1.3],
       [6.9, 3.1, 5.1, 2.3],
       [6.2, 2.2, 4.5, 1.5],
       [5.8, 2.7, 3.9, 1.2],
       [6.5, 3.2, 5.1, 2. ],
       [4.8, 3. , 1.4, 0.1],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.1, 3.8, 1.5, 0.3],
       [6.3, 3.3, 4.7, 1.6],
       [6.5, 3. , 5.8, 2.2],
       [5.6, 2.5, 3.9, 1.1],
       [5.7, 2.8, 4.5, 1.3],
       [6.4, 2.8, 5.6, 2.2],
       [4.7, 3.2, 1.6, 0.2],
       [6.1, 3. , 4.9, 1.8],
       [5. , 3.4, 1.6, 0.4],
       [6.4, 2.8, 5.6, 2.1],
       [7.9, 3.8, 6.4, 2. ],
       [6.7, 3. , 5.2, 2.3],
       [6.7, 2.5, 5.8, 1.8],
       [6.8, 3.2, 5.9, 2.3],
       [4.8, 3. , 1.4, 0.3],
       [4.8, 3.1, 1.6, 0.2],
       [4.6, 3.6, 1. , 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [6.7, 3.1, 4.4, 1.4],
       [4.8, 3.4, 1.6, 0.2],
       [4.4, 3.2, 1.3, 0.2],
       [6.3, 2.5, 5. , 1.9],
       [6.4, 3.2, 4.5, 1.5],
       [5.2, 3.5, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.2, 4.1, 1.5, 0.1],
       [5.8, 2.7, 5.1, 1.9],
```

```
       [6. , 3.4, 4.5, 1.6],
       [6.7, 3.1, 4.7, 1.5],
       [5.4, 3.9, 1.3, 0.4],
       [5.4, 3.7, 1.5, 0.2],
       [5.5, 2.4, 3.7, 1. ],
       [6.3, 2.8, 5.1, 1.5],
       [6.4, 3.1, 5.5, 1.8],
       [6.6, 3. , 4.4, 1.4],
       [7.2, 3.6, 6.1, 2.5]])
```

In [24]:
```
y_train
```

Out[24]:
```
array([[0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
```

```
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
```

```
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]], dtype=float32)
```

In [25]:
```
y_test
```

Out[25]:
```
array([[0., 1., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
```

```
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.]], dtype=float32)
```

Observation: We see train_test_split does the shuffling of rows of data. Note: We don't want the rows in order as given in the dataset so that learning can be done on all 3 classes of y (i.e. all 3 labels)

Standardizing the Data

In [30]:
```
from sklearn.preprocessing import MinMaxScaler
```

The minmax scale will do standardization which is just normalizing the values to fit between a certain range 0 to 1, or -1 to 1. In this case, we will divide by the max value in the array to get all the array values between 0 and 1.

In [31]:
```
# Example
np.array([5, 10, 15, 20])/20
```

Out[31]:
```
array([0.25, 0.5 , 0.75, 1.  ])
```

In [32]:
```
scaler_object = MinMaxScaler()
```

In [33]:
```
# Fit the scaler_object with the X_train data
scaler_object.fit(X_train)
```

Out[33]:
```
MinMaxScaler(copy=True, feature_range=(0, 1))
```

In [34]:
```
scaled_X_train = scaler_object.fit_transform(X_train)
```

In [35]:
```
scaled_X_test = scaler_object.fit_transform(X_test)
```

In [36]:
```
scaled_X_train
```

Out[36]:
```
array([[0.41176471, 0.40909091, 0.55357143, 0.5       ],
       [0.97058824, 0.45454545, 0.98214286, 0.83333333],
       [0.38235294, 0.45454545, 0.60714286, 0.58333333],
       [0.23529412, 0.68181818, 0.05357143, 0.04166667],
       [1.        , 0.36363636, 1.        , 0.79166667],
       [0.44117647, 0.31818182, 0.53571429, 0.375     ],
       [0.26470588, 0.63636364, 0.05357143, 0.04166667],
       [0.20588235, 0.68181818, 0.03571429, 0.08333333],
       [0.23529412, 0.81818182, 0.14285714, 0.125     ],
       [0.20588235, 0.        , 0.42857143, 0.375     ],
       [0.58823529, 0.31818182, 0.67857143, 0.70833333],
       [0.14705882, 0.63636364, 0.14285714, 0.04166667],
       [0.20588235, 0.45454545, 0.08928571, 0.04166667],
       [0.23529412, 0.59090909, 0.10714286, 0.16666667],
       [0.38235294, 0.31818182, 0.55357143, 0.5       ],
       [0.23529412, 0.63636364, 0.07142857, 0.04166667],
       [0.41176471, 0.45454545, 0.55357143, 0.45833333],
       [1.        , 0.81818182, 1.        , 0.875     ],
       [0.08823529, 0.54545455, 0.05357143, 0.04166667],
       [0.55882353, 0.40909091, 0.57142857, 0.5       ],
       [0.41176471, 0.22727273, 0.69642857, 0.79166667],
       [0.35294118, 1.        , 0.05357143, 0.04166667],
       [0.5       , 0.45454545, 0.66071429, 0.70833333],
       [0.44117647, 0.31818182, 0.71428571, 0.75      ],
       [0.5       , 0.09090909, 0.51785714, 0.375     ],
       [0.32352941, 0.45454545, 0.60714286, 0.58333333],
       [0.55882353, 0.63636364, 0.76785714, 0.91666667],
       [0.35294118, 0.13636364, 0.51785714, 0.5       ],
       [0.32352941, 0.86363636, 0.10714286, 0.125     ],
       [0.20588235, 0.13636364, 0.39285714, 0.375     ],
       [0.61764706, 0.31818182, 0.75      , 0.75      ],
       [0.20588235, 0.59090909, 0.05357143, 0.04166667],
       [0.20588235, 0.54545455, 0.01785714, 0.04166667],
       [0.35294118, 0.18181818, 0.48214286, 0.41666667],
       [0.70588235, 0.45454545, 0.69642857, 0.66666667],
       [0.17647059, 0.5       , 0.07142857, 0.04166667],
       [0.44117647, 0.36363636, 0.71428571, 0.95833333],
       [0.20588235, 0.63636364, 0.07142857, 0.04166667],
       [0.20588235, 0.68181818, 0.08928571, 0.20833333],
       [0.47058824, 0.54545455, 0.66071429, 0.70833333],
       [0.23529412, 0.22727273, 0.33928571, 0.41666667],
       [0.76470588, 0.54545455, 0.82142857, 0.91666667],
       [0.5       , 0.31818182, 0.71428571, 0.625     ],
       [0.52941176, 0.27272727, 0.80357143, 0.54166667],
       [1.        , 0.45454545, 0.89285714, 0.91666667],
       [0.35294118, 0.22727273, 0.51785714, 0.5       ],
       [0.02941176, 0.40909091, 0.05357143, 0.04166667],
       [0.        , 0.45454545, 0.        , 0.        ],
```

```
[0.5       , 0.09090909, 0.69642857, 0.58333333],
[0.85294118, 0.54545455, 0.875     , 0.70833333],
[0.08823529, 0.5       , 0.07142857, 0.04166667],
[0.23529412, 0.68181818, 0.05357143, 0.08333333],
[0.02941176, 0.45454545, 0.03571429, 0.04166667],
[0.58823529, 0.22727273, 0.67857143, 0.58333333],
[0.58823529, 0.63636364, 0.80357143, 0.95833333],
[0.08823529, 0.63636364, 0.05357143, 0.08333333],
[0.73529412, 0.45454545, 0.78571429, 0.83333333],
[0.58823529, 0.59090909, 0.875     , 1.        ],
[0.11764706, 0.54545455, 0.03571429, 0.04166667],
[0.52941176, 0.40909091, 0.64285714, 0.54166667],
[0.64705882, 0.36363636, 0.625     , 0.58333333],
[0.55882353, 0.36363636, 0.66071429, 0.70833333],
[0.79411765, 0.54545455, 0.64285714, 0.54166667],
[0.61764706, 0.54545455, 0.75      , 0.91666667],
[0.23529412, 0.81818182, 0.08928571, 0.04166667],
[0.76470588, 0.5       , 0.76785714, 0.83333333],
[0.47058824, 0.45454545, 0.55357143, 0.58333333],
[0.64705882, 0.45454545, 0.73214286, 0.79166667],
[0.41176471, 0.27272727, 0.42857143, 0.375     ],
[0.26470588, 0.31818182, 0.5       , 0.54166667],
[0.52941176, 0.45454545, 0.625     , 0.54166667],
[0.05882353, 0.13636364, 0.03571429, 0.08333333],
[0.67647059, 0.40909091, 0.625     , 0.5       ],
[0.35294118, 0.27272727, 0.58928571, 0.45833333],
[0.29411765, 0.77272727, 0.07142857, 0.04166667],
[0.38235294, 0.45454545, 0.53571429, 0.5       ],
[0.88235294, 0.40909091, 0.92857143, 0.70833333],
[0.70588235, 0.59090909, 0.82142857, 0.83333333],
[0.23529412, 0.77272727, 0.07142857, 0.125     ],
[0.17647059, 0.18181818, 0.39285714, 0.375     ],
[0.70588235, 0.59090909, 0.82142857, 1.        ],
[0.85294118, 0.45454545, 0.83928571, 0.625     ],
[0.17647059, 0.72727273, 0.05357143, 0.        ],
[0.70588235, 0.5       , 0.80357143, 0.95833333],
[0.17647059, 0.45454545, 0.05357143, 0.04166667],
[0.76470588, 0.5       , 0.67857143, 0.58333333],
[0.91176471, 0.36363636, 0.89285714, 0.75      ],
[0.58823529, 0.40909091, 0.80357143, 0.70833333],
[0.41176471, 0.36363636, 0.53571429, 0.5       ],
[0.64705882, 0.45454545, 0.78571429, 0.70833333],
[0.58823529, 0.13636364, 0.58928571, 0.5       ],
[0.61764706, 0.40909091, 0.57142857, 0.5       ],
[0.38235294, 0.36363636, 0.67857143, 0.79166667],
[0.47058824, 0.45454545, 0.71428571, 0.70833333],
[0.32352941, 0.63636364, 0.10714286, 0.04166667],
[0.52941176, 0.36363636, 0.51785714, 0.5       ],
[0.17647059, 0.22727273, 0.60714286, 0.66666667],
[0.44117647, 0.90909091, 0.01785714, 0.04166667],
```

```
       [0.44117647, 0.27272727, 0.51785714, 0.45833333],
       [0.82352941, 0.45454545, 0.85714286, 0.83333333]])
```

In [37]:
```
scaled_X_test
```

Out[37]:
```
array([[0.48571429, 0.27272727, 0.62711864, 0.45833333],
       [0.37142857, 0.72727273, 0.11864407, 0.08333333],
       [0.94285714, 0.18181818, 1.        , 0.91666667],
       [0.45714286, 0.31818182, 0.59322034, 0.58333333],
       [0.68571429, 0.27272727, 0.6440678 , 0.54166667],
       [0.28571429, 0.54545455, 0.08474576, 0.125     ],
       [0.34285714, 0.31818182, 0.44067797, 0.5       ],
       [0.71428571, 0.40909091, 0.69491525, 0.91666667],
       [0.51428571, 0.        , 0.59322034, 0.58333333],
       [0.4       , 0.22727273, 0.49152542, 0.45833333],
       [0.6       , 0.45454545, 0.69491525, 0.79166667],
       [0.11428571, 0.36363636, 0.06779661, 0.        ],
       [0.31428571, 0.59090909, 0.05084746, 0.04166667],
       [0.14285714, 0.40909091, 0.08474576, 0.        ],
       [0.2       , 0.72727273, 0.08474576, 0.08333333],
       [0.54285714, 0.5       , 0.62711864, 0.625     ],
       [0.6       , 0.36363636, 0.81355932, 0.875     ],
       [0.34285714, 0.13636364, 0.49152542, 0.41666667],
       [0.37142857, 0.27272727, 0.59322034, 0.5       ],
       [0.57142857, 0.27272727, 0.77966102, 0.875     ],
       [0.08571429, 0.45454545, 0.10169492, 0.04166667],
       [0.48571429, 0.36363636, 0.66101695, 0.70833333],
       [0.17142857, 0.54545455, 0.10169492, 0.125     ],
       [0.57142857, 0.27272727, 0.77966102, 0.83333333],
       [1.        , 0.72727273, 0.91525424, 0.79166667],
       [0.65714286, 0.36363636, 0.71186441, 0.91666667],
       [0.65714286, 0.13636364, 0.81355932, 0.70833333],
       [0.68571429, 0.45454545, 0.83050847, 0.91666667],
       [0.11428571, 0.36363636, 0.06779661, 0.08333333],
       [0.11428571, 0.40909091, 0.10169492, 0.04166667],
       [0.05714286, 0.63636364, 0.        , 0.04166667],
       [0.37142857, 1.        , 0.08474576, 0.125     ],
       [0.65714286, 0.40909091, 0.57627119, 0.54166667],
       [0.11428571, 0.54545455, 0.10169492, 0.04166667],
       [0.        , 0.45454545, 0.05084746, 0.04166667],
       [0.54285714, 0.13636364, 0.6779661 , 0.75      ],
       [0.57142857, 0.45454545, 0.59322034, 0.58333333],
       [0.22857143, 0.59090909, 0.08474576, 0.04166667],
       [0.17142857, 0.63636364, 0.06779661, 0.04166667],
       [0.22857143, 0.86363636, 0.08474576, 0.        ],
       [0.4       , 0.22727273, 0.69491525, 0.75      ],
       [0.45714286, 0.54545455, 0.59322034, 0.625     ],
       [0.65714286, 0.40909091, 0.62711864, 0.58333333],
       [0.28571429, 0.77272727, 0.05084746, 0.125     ],
       [0.28571429, 0.68181818, 0.08474576, 0.04166667],
```

```
       [0.31428571, 0.09090909, 0.45762712, 0.375     ],
       [0.54285714, 0.27272727, 0.69491525, 0.58333333],
       [0.57142857, 0.40909091, 0.76271186, 0.70833333],
       [0.62857143, 0.36363636, 0.57627119, 0.54166667],
       [0.8       , 0.63636364, 0.86440678, 1.        ]])
```

Observation: Now scaled_X_train and scaled_X_test have values 0 to 1.

In [38]:
```
X_train.max()
```

Out[38]: 7.7

In [39]:
```
scaled_X_train.max()
```

Out[39]: 1.0

Building the Network with Keras

We will build a simple neural network

In [40]:
```
from keras.models import Sequential
from keras.layers import Dense
```

In [41]:
```
model = Sequential()
model.add(Dense(8, input_dim = 4, activation= 'relu'))
# Number of neurons in this layer is 8. It should be a multiple of the
# input_dim. "relu" = Rectified Linear Unit.
model.add(Dense(8, input_dim = 4, activation = 'relu'))
model.add(Dense (3, activation = 'softmax'))
# Output layer has 3 neurons for the 3 classes and activation = 'softmax
# It is actually the probability of belonging to a particular class
# Suppose y = [0, 0, 1], we may get our predicted y = [0.2, 0.3, 0.5]
# i.e. probability is 0.5 that the observation belongs to class 3
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam',
              metrics = ['accuracy'])
```

```
WARNING:tensorflow:From /Users/jayashrijagannathan/anaconda3/envs/nlp_
course/lib/python3.7/site-packages/tensorflow/python/framework/op_def_
library.py:263: colocate_with (from tensorflow.python.framework.ops) i
s deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

In [43]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 8)                 40
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 72
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 27
=================================================================
Total params: 139
Trainable params: 139
Non-trainable params: 0
_________________________________________________________________
```

Fit (Train) the Model

In [44]:
```
model.fit(scaled_X_train, y_train, epochs = 150, verbose = 2)
```

```
WARNING:tensorflow:From /Users/jayashrijagannathan/anaconda3/envs/nlp
_course/lib/python3.7/site-packages/tensorflow/python/ops/math_ops.py
:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated a
nd will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/150
 - 1s - loss: 1.0992 - acc: 0.3500
Epoch 2/150
 - 0s - loss: 1.0947 - acc: 0.3500
Epoch 3/150
 - 0s - loss: 1.0904 - acc: 0.3500
Epoch 4/150
 - 0s - loss: 1.0863 - acc: 0.3500
Epoch 5/150
 - 0s - loss: 1.0823 - acc: 0.3500
Epoch 6/150
 - 0s - loss: 1.0782 - acc: 0.3500
Epoch 7/150
```

Predicting New Unseen Data

In [45]:
```
scaled_X_test
```

Out[45]:
```
array([[0.48571429, 0.27272727, 0.62711864, 0.45833333],
       [0.37142857, 0.72727273, 0.11864407, 0.08333333],
```

```
       [0.94285714, 0.18181818, 1.        , 0.91666667],
       [0.45714286, 0.31818182, 0.59322034, 0.58333333],
       [0.68571429, 0.27272727, 0.6440678 , 0.54166667],
       [0.28571429, 0.54545455, 0.08474576, 0.125     ],
       [0.34285714, 0.31818182, 0.44067797, 0.5       ],
       [0.71428571, 0.40909091, 0.69491525, 0.91666667],
       [0.51428571, 0.        , 0.59322034, 0.58333333],
       [0.4       , 0.22727273, 0.49152542, 0.45833333],
       [0.6       , 0.45454545, 0.69491525, 0.79166667],
       [0.11428571, 0.36363636, 0.06779661, 0.        ],
       [0.31428571, 0.59090909, 0.05084746, 0.04166667],
       [0.14285714, 0.40909091, 0.08474576, 0.        ],
       [0.2       , 0.72727273, 0.08474576, 0.08333333],
       [0.54285714, 0.5       , 0.62711864, 0.625     ],
       [0.6       , 0.36363636, 0.81355932, 0.875     ],
       [0.34285714, 0.13636364, 0.49152542, 0.41666667],
       [0.37142857, 0.27272727, 0.59322034, 0.5       ],
       [0.57142857, 0.27272727, 0.77966102, 0.875     ],
       [0.08571429, 0.45454545, 0.10169492, 0.04166667],
       [0.48571429, 0.36363636, 0.66101695, 0.70833333],
       [0.17142857, 0.54545455, 0.10169492, 0.125     ],
       [0.57142857, 0.27272727, 0.77966102, 0.83333333],
       [1.        , 0.72727273, 0.91525424, 0.79166667],
       [0.65714286, 0.36363636, 0.71186441, 0.91666667],
       [0.65714286, 0.13636364, 0.81355932, 0.70833333],
       [0.68571429, 0.45454545, 0.83050847, 0.91666667],
       [0.11428571, 0.36363636, 0.06779661, 0.08333333],
       [0.11428571, 0.40909091, 0.10169492, 0.04166667],
       [0.05714286, 0.63636364, 0.        , 0.04166667],
       [0.37142857, 1.        , 0.08474576, 0.125     ],
       [0.65714286, 0.40909091, 0.57627119, 0.54166667],
       [0.11428571, 0.54545455, 0.10169492, 0.04166667],
       [0.        , 0.45454545, 0.05084746, 0.04166667],
       [0.54285714, 0.13636364, 0.6779661 , 0.75      ],
       [0.57142857, 0.45454545, 0.59322034, 0.58333333],
       [0.22857143, 0.59090909, 0.08474576, 0.04166667],
       [0.17142857, 0.63636364, 0.06779661, 0.04166667],
       [0.22857143, 0.86363636, 0.08474576, 0.        ],
       [0.4       , 0.22727273, 0.69491525, 0.75      ],
       [0.45714286, 0.54545455, 0.59322034, 0.625     ],
       [0.65714286, 0.40909091, 0.62711864, 0.58333333],
       [0.28571429, 0.77272727, 0.05084746, 0.125     ],
       [0.28571429, 0.68181818, 0.08474576, 0.04166667],
       [0.31428571, 0.09090909, 0.45762712, 0.375     ],
       [0.54285714, 0.27272727, 0.69491525, 0.58333333],
       [0.57142857, 0.40909091, 0.76271186, 0.70833333],
       [0.62857143, 0.36363636, 0.57627119, 0.54166667],
       [0.8       , 0.63636364, 0.86440678, 1.        ]])
```

In [46]:
```
model.predict(scaled X test)
```

Out[46]:
```
array([[8.65308382e-03, 4.67338741e-01, 5.24008155e-01],
       [9.54618752e-01, 3.59401964e-02, 9.44108237e-03],
       [4.44544239e-05, 2.88910210e-01, 7.11045325e-01],
       [8.18765815e-03, 4.69791114e-01, 5.22021174e-01],
       [2.27987487e-03, 4.17570591e-01, 5.80149591e-01],
       [9.29278791e-01, 5.44792265e-02, 1.62420012e-02],
       [5.52858301e-02, 5.26129782e-01, 4.18584347e-01],
       [4.30612068e-04, 3.64578515e-01, 6.34990871e-01],
       [1.79621659e-03, 3.95891279e-01, 6.02312565e-01],
       [2.39778366e-02, 5.00243127e-01, 4.75778967e-01],
       [1.64045661e-03, 4.18262750e-01, 5.80096781e-01],
       [9.27865446e-01, 5.46479672e-02, 1.74865518e-02],
       [9.56540406e-01, 3.41143198e-02, 9.34518408e-03],
       [9.33221221e-01, 5.08760363e-02, 1.59027800e-02],
       [9.68274474e-01, 2.53733862e-02, 6.35219738e-03],
       [1.04111414e-02, 4.93940055e-01, 4.95648801e-01],
       [2.60298548e-04, 3.39304417e-01, 6.60435319e-01],
       [2.22768448e-02, 4.89968181e-01, 4.87754941e-01],
       [1.36149507e-02, 4.83498633e-01, 5.02886355e-01],
       [3.09920200e-04, 3.43725920e-01, 6.55964077e-01],
       [9.37970936e-01, 4.75069284e-02, 1.45221548e-02],
       [3.01533937e-03, 4.34255570e-01, 5.62729061e-01],
       [9.40481722e-01, 4.59773056e-02, 1.35409599e-02],
       [3.53768963e-04, 3.47607851e-01, 6.52038395e-01],
       [3.42983956e-04, 3.78166348e-01, 6.21490657e-01],
       [3.75019881e-04, 3.55311781e-01, 6.44313276e-01],
       [3.69335059e-04, 3.48369807e-01, 6.51260793e-01],
       [2.23382041e-04, 3.40895355e-01, 6.58881307e-01],
       [9.21930909e-01, 5.88565245e-02, 1.92125682e-02],
       [9.29694176e-01, 5.33632562e-02, 1.69426911e-02],
       [9.66602981e-01, 2.66287122e-02, 6.76828437e-03],
       [9.82831895e-01, 1.42107112e-02, 2.95742694e-03],
       [8.97926744e-03, 4.82745796e-01, 5.08274913e-01],
       [9.50479925e-01, 3.85159366e-02, 1.10041928e-02],
       [9.43622231e-01, 4.34671640e-02, 1.29106585e-02],
       [7.21842225e-04, 3.68513495e-01, 6.30764663e-01],
       [1.14368657e-02, 4.94376719e-01, 4.94186401e-01],
       [9.55698490e-01, 3.47088650e-02, 9.59267840e-03],
       [9.62523937e-01, 2.96725724e-02, 7.80342100e-03],
       [9.78744864e-01, 1.73862465e-02, 3.86886997e-03],
       [1.20949023e-03, 3.86808783e-01, 6.11981690e-01],
       [2.34179683e-02, 5.24613917e-01, 4.51968074e-01],
       [5.02181053e-03, 4.59723115e-01, 5.35255134e-01],
       [9.71884012e-01, 2.26450227e-02, 5.47087099e-03],
       [9.64398265e-01, 2.82773543e-02, 7.32442969e-03],
       [3.02295629e-02, 4.95239943e-01, 4.74530488e-01],
       [2.19698297e-03, 4.14037555e-01, 5.83765388e-01],
       [1.35400600e-03, 4.05348241e-01, 5.93297720e-01],
       [7.74421683e-03, 4.73191947e-01, 5.19063830e-01],
```

```
       [2.05775417e-04, 3.51861209e-01, 6.47932947e-01]], dtype=float3
2)
```

Observation: This gives us probabilities by default. If we want the raw classes themselves, we will call predict_classes as given below.

In [47]:
```
model.predict_classes(scaled_X_test)
```

Out[47]:
```
array([2, 0, 2, 2, 2, 0, 1, 2, 2, 1, 2, 0, 0, 0, 0, 2, 2, 1, 2, 2, 0,
2,
       0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 2, 0, 0, 2, 1, 0, 0, 0, 2, 1, 2,
0,
       0, 1, 2, 2, 2, 2])
```

In [48]:
```
y_test
```

Out[48]:
```
array([[0., 1., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
```

```
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 1., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.]], dtype=float32)
```

In [49]:
```
# We need to compare the predictions to y_test
# We will change y_test from One Hot encoding to give us the
# index of the max value

y_test.argmax(axis = 1)
```

Out[49]:
```
array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0,
2,
       0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1,
0,
       0, 1, 2, 2, 1, 2])
```

Evaluating Model Performance

In [50]:
```
model.metrics_names
```

Out[50]:
```
['loss', 'acc']
```

In [51]:
```
model.evaluate(x = scaled_X_test, y = y_test)
```

```
50/50 [==============================] - 0s 3ms/step
```

Out[51]:
```
[0.3887695848941803, 0.8199999928474426]
```

In [52]:
```
from sklearn.metrics import confusion_matrix, classification_report
```

In [55]:
```
predictions = model.predict_classes(scaled_X_test)
```

In [56]:
```
predictions
```

Out[56]:
```
array([2, 0, 2, 2, 2, 0, 1, 2, 2, 1, 2, 0, 0, 0, 0, 2, 2, 1, 2, 2, 0,
2,
       0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 2, 0, 0, 2, 1, 0, 0, 0, 2, 1, 2,
0,
       0, 1, 2, 2, 2, 2])
```

In [57]:
```
y_test.argmax(axis = 1)
```

Out[57]:
```
array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0,
2,
       0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 2, 1, 1,
0,
       0, 1, 2, 2, 1, 2])
```

In [58]:
```
confusion_matrix(y_test.argmax(axis = 1), predictions)
```

Out[58]:
```
array([[19,  0,  0],
       [ 0,  6,  9],
       [ 0,  0, 16]])
```

In [61]:
```
print(classification_report(y_test.argmax(axis = 1), predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.40      0.57        15
           2       0.64      1.00      0.78        16

   micro avg       0.82      0.82      0.82        50
   macro avg       0.88      0.80      0.78        50
weighted avg       0.88      0.82      0.80        50
```

In [67]:
```
from sklearn.metrics import accuracy_score

print(accuracy_score(y_test.argmax(axis = 1), predictions))
```

```
0.82
```

Saving and Loading Models

In [62]:
```
# Save the model
model.save('myfirstmodel_JJ.h5')
```

In [63]:
```
from keras.models import load_model
```

In [64]:
```
new_model = load_model('myfirstmodel_JJ.h5')
```

In [68]:
```
new_model.predict_classes(scaled_X_test)
```

Out[68]:
```
array([2, 0, 2, 2, 2, 0, 1, 2, 2, 1, 2, 0, 0, 0, 0, 2, 2, 1, 2, 2, 0,
2,
       0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 2, 0, 0, 2, 1, 0, 0, 0, 2, 1, 2,
0,
       0, 1, 2, 2, 2, 2])
```

In [69]:
```
model.predict_classes(scaled_X_test)
```

Out[69]:
```
array([2, 0, 2, 2, 2, 0, 1, 2, 2, 1, 2, 0, 0, 0, 0, 2, 2, 1, 2, 2, 0,
2,
       0, 2, 2, 2, 2, 2, 0, 0, 0, 0, 2, 0, 0, 2, 1, 0, 0, 0, 2, 1, 2,
0,
       0, 1, 2, 2, 2, 2])
```

In [ ]:
```
# Comparing that new_model and model are the same.
```