```python
#
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, f1_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

data = pd.read_csv('Iris.csv') # Importing the dataset

data
```

```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
```

```
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
data.isnull().sum()
```

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
data = data.drop(["Id"], axis="columns")   # removing the unwanted
columns
```

```
data
```

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
Species
0              5.1           3.5            1.4           0.2
Iris-setosa
1              4.9           3.0            1.4           0.2
Iris-setosa
2              4.7           3.2            1.3           0.2
Iris-setosa
3              4.6           3.1            1.5           0.2
Iris-setosa
4              5.0           3.6            1.4           0.2
Iris-setosa
..             ...           ...            ...           ...
...
145            6.7           3.0            5.2           2.3  Iris-
virginica
146            6.3           2.5            5.0           1.9  Iris-
virginica
147            6.5           3.0            5.2           2.0  Iris-
virginica
148            6.2           3.4            5.4           2.3  Iris-
virginica
149            5.9           3.0            5.1           1.8  Iris-
virginica

[150 rows x 5 columns]
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
 4   Species        150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

```
data["Species"].unique()
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'],
dtype=object)
```

```
from sklearn import preprocessing          #applying lable-encoder to
convert categorical variable into numerical.
Encode = preprocessing.LabelEncoder()
Encode.fit(['Iris-setosa','Iris-versicolor','Iris-virginica'])
data["Species"] = Encode.transform(data['Species'])
```

```
data["Species"].unique()
```

```
array([0, 1, 2])
```

```
data
```

```
     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
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
..             ...           ...            ...           ...      ...
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 x 5 columns]
```

```
x = data.drop("Species", axis= 1)
x = np.asanyarray(x)
x
```

```
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],
[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, 0.1],
[5. , 3.2, 1.2, 0.2],
[5.5, 3.5, 1.3, 0.2],
[4.9, 3.1, 1.5, 0.1],
[4.4, 3. , 1.3, 0.2],
[5.1, 3.4, 1.5, 0.2],
[5. , 3.5, 1.3, 0.3],
[4.5, 2.3, 1.3, 0.3],
[4.4, 3.2, 1.3, 0.2],
[5. , 3.5, 1.6, 0.6],
[5.1, 3.8, 1.9, 0.4],
[4.8, 3. , 1.4, 0.3],
[5.1, 3.8, 1.6, 0.2],
[4.6, 3.2, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[5. , 3.3, 1.4, 0.2],
[7. , 3.2, 4.7, 1.4],
[6.4, 3.2, 4.5, 1.5],
[6.9, 3.1, 4.9, 1.5],
```

```
[5.5, 2.3, 4. , 1.3],
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
```

```
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

```python
y = np.asanyarray(data["Species"])    #Dependent veriable
y
```

```
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])
```

```python
from sklearn.preprocessing import normalize
x_nor = normalize(x,axis=0)
x_nor
```

```
array([[0.07056264, 0.09265065, 0.02754646, 0.01150299],
       [0.06779548, 0.07941484, 0.02754646, 0.01150299],
       [0.06502832, 0.08470916, 0.02557886, 0.01150299],
       [0.06364474, 0.082062  , 0.02951407, 0.01150299],
       [0.06917906, 0.09529781, 0.02754646, 0.01150299],
       [0.07471338, 0.10323929, 0.03344928, 0.02300599],
       [0.06364474, 0.09000348, 0.02754646, 0.01725449],
       [0.06917906, 0.09000348, 0.02951407, 0.01150299],
       [0.06087757, 0.07676768, 0.02754646, 0.01150299],
       [0.06779548, 0.082062  , 0.02951407, 0.0057515 ],
       [0.07471338, 0.09794497, 0.02951407, 0.01150299],
       [0.0664119 , 0.09000348, 0.03148167, 0.01150299],
       [0.0664119 , 0.07941484, 0.02754646, 0.0057515 ],
       [0.05949399, 0.07941484, 0.02164365, 0.0057515 ],
       [0.08024771, 0.10588645, 0.02361125, 0.01150299],
       [0.07886413, 0.1164751 , 0.02951407, 0.02300599],
       [0.07471338, 0.10323929, 0.02557886, 0.02300599],
       [0.07056264, 0.09265065, 0.02754646, 0.01725449],
       [0.07886413, 0.10059213, 0.03344928, 0.01725449],
       [0.07056264, 0.10059213, 0.02951407, 0.01725449],
       [0.07471338, 0.09000348, 0.03344928, 0.01150299],
       [0.07056264, 0.09794497, 0.02951407, 0.02300599],
       [0.06364474, 0.09529781, 0.01967604, 0.01150299],
       [0.07056264, 0.08735632, 0.03344928, 0.02875749],
       [0.0664119 , 0.09000348, 0.03738448, 0.01150299],
       [0.06917906, 0.07941484, 0.03148167, 0.01150299],
       [0.06917906, 0.09000348, 0.03148167, 0.02300599],
       [0.07194622, 0.09265065, 0.02951407, 0.01150299],
       [0.07194622, 0.09000348, 0.02754646, 0.01150299],
       [0.06502832, 0.08470916, 0.03148167, 0.01150299],
       [0.0664119 , 0.082062  , 0.03148167, 0.01150299],
       [0.07471338, 0.09000348, 0.02951407, 0.02300599],
       [0.07194622, 0.10853361, 0.02951407, 0.0057515 ],
```

```
[0.07609697, 0.11118077, 0.02754646, 0.01150299],
[0.06779548, 0.082062  , 0.02951407, 0.0057515 ],
[0.06917906, 0.08470916, 0.02361125, 0.01150299],
[0.07609697, 0.09265065, 0.02557886, 0.01150299],
[0.06779548, 0.082062  , 0.02951407, 0.0057515 ],
[0.06087757, 0.07941484, 0.02557886, 0.01150299],
[0.07056264, 0.09000348, 0.02951407, 0.01150299],
[0.06917906, 0.09265065, 0.02557886, 0.01725449],
[0.06226115, 0.06088471, 0.02557886, 0.01725449],
[0.06087757, 0.08470916, 0.02557886, 0.01150299],
[0.06917906, 0.09265065, 0.03148167, 0.03450898],
[0.07056264, 0.10059213, 0.03738448, 0.02300599],
[0.0664119 , 0.07941484, 0.02754646, 0.01725449],
[0.07056264, 0.10059213, 0.03148167, 0.01150299],
[0.06364474, 0.08470916, 0.02754646, 0.01150299],
[0.0733298 , 0.09794497, 0.02951407, 0.01150299],
[0.06917906, 0.08735632, 0.02754646, 0.01150299],
[0.09685068, 0.08470916, 0.09247741, 0.08052096],
[0.0885492 , 0.08470916, 0.0885422 , 0.08627246],
[0.0954671 , 0.082062  , 0.09641262, 0.08627246],
[0.07609697, 0.06088471, 0.07870418, 0.07476947],
[0.08993278, 0.07412052, 0.0905098 , 0.08627246],
[0.07886413, 0.07412052, 0.0885422 , 0.07476947],
[0.08716562, 0.08735632, 0.09247741, 0.09202396],
[0.06779548, 0.06353187, 0.06493095, 0.05751497],
[0.09131636, 0.07676768, 0.0905098 , 0.07476947],
[0.07194622, 0.07147336, 0.07673657, 0.08052096],
[0.06917906, 0.05294323, 0.06886616, 0.05751497],
[0.08163129, 0.07941484, 0.08263939, 0.08627246],
[0.08301487, 0.05823755, 0.07870418, 0.05751497],
[0.08439845, 0.07676768, 0.09247741, 0.08052096],
[0.07748055, 0.07676768, 0.07083376, 0.07476947],
[0.09269994, 0.082062  , 0.0865746 , 0.08052096],
[0.07748055, 0.07941484, 0.0885422 , 0.08627246],
[0.08024771, 0.07147336, 0.08067178, 0.05751497],
[0.08578203, 0.05823755, 0.0885422 , 0.08627246],
[0.07748055, 0.06617903, 0.07673657, 0.06326647],
[0.08163129, 0.08470916, 0.09444501, 0.10352695],
[0.08439845, 0.07412052, 0.07870418, 0.07476947],
[0.08716562, 0.06617903, 0.09641262, 0.08627246],
[0.08439845, 0.07412052, 0.09247741, 0.06901797],
[0.0885492 , 0.07676768, 0.08460699, 0.07476947],
[0.09131636, 0.07941484, 0.0865746 , 0.08052096],
[0.09408352, 0.07412052, 0.09444501, 0.08052096],
[0.09269994, 0.07941484, 0.09838022, 0.09777546],
[0.08301487, 0.07676768, 0.0885422 , 0.08627246],
[0.07886413, 0.06882619, 0.06886616, 0.05751497],
[0.07609697, 0.06353187, 0.07476897, 0.06326647],
[0.07609697, 0.06353187, 0.07280136, 0.05751497],
[0.08024771, 0.07147336, 0.07673657, 0.06901797],
```

```
[0.08301487, 0.07147336, 0.10034783, 0.09202396],
[0.07471338, 0.07941484, 0.0885422 , 0.08627246],
[0.08301487, 0.09000348, 0.0885422 , 0.09202396],
[0.09269994, 0.082062  , 0.09247741, 0.08627246],
[0.08716562, 0.06088471, 0.0865746 , 0.07476947],
[0.07748055, 0.07941484, 0.08067178, 0.07476947],
[0.07609697, 0.06617903, 0.07870418, 0.07476947],
[0.07609697, 0.06882619, 0.0865746 , 0.06901797],
[0.08439845, 0.07941484, 0.0905098 , 0.08052096],
[0.08024771, 0.06882619, 0.07870418, 0.06901797],
[0.06917906, 0.06088471, 0.06493095, 0.05751497],
[0.07748055, 0.07147336, 0.08263939, 0.07476947],
[0.07886413, 0.07941484, 0.08263939, 0.06901797],
[0.07886413, 0.07676768, 0.08263939, 0.07476947],
[0.08578203, 0.07676768, 0.08460699, 0.07476947],
[0.07056264, 0.06617903, 0.05902813, 0.06326647],
[0.07886413, 0.07412052, 0.08067178, 0.07476947],
[0.08716562, 0.08735632, 0.11805627, 0.14378743],
[0.08024771, 0.07147336, 0.10034783, 0.10927845],
[0.09823426, 0.07941484, 0.11608866, 0.12078145],
[0.08716562, 0.07676768, 0.11018585, 0.10352695],
[0.08993278, 0.07941484, 0.11412106, 0.12653294],
[0.10515217, 0.07941484, 0.12986189, 0.12078145],
[0.06779548, 0.06617903, 0.0885422 , 0.09777546],
[0.10100143, 0.07676768, 0.12395908, 0.10352695],
[0.09269994, 0.06617903, 0.11412106, 0.10352695],
[0.09961785, 0.09529781, 0.12002387, 0.14378743],
[0.08993278, 0.08470916, 0.10034783, 0.11502995],
[0.0885492 , 0.07147336, 0.10428304, 0.10927845],
[0.09408352, 0.07941484, 0.10821824, 0.12078145],
[0.07886413, 0.06617903, 0.09838022, 0.11502995],
[0.08024771, 0.07412052, 0.10034783, 0.13803594],
[0.0885492 , 0.08470916, 0.10428304, 0.13228444],
[0.08993278, 0.07941484, 0.10821824, 0.10352695],
[0.10653575, 0.10059213, 0.1318295 , 0.12653294],
[0.10653575, 0.06882619, 0.13576471, 0.13228444],
[0.08301487, 0.05823755, 0.09838022, 0.08627246],
[0.0954671 , 0.08470916, 0.11215345, 0.13228444],
[0.07748055, 0.07412052, 0.09641262, 0.11502995],
[0.10653575, 0.07412052, 0.1318295 , 0.11502995],
[0.08716562, 0.07147336, 0.09641262, 0.10352695],
[0.09269994, 0.08735632, 0.11215345, 0.12078145],
[0.09961785, 0.08470916, 0.11805627, 0.10352695],
[0.08578203, 0.07412052, 0.09444501, 0.10352695],
[0.08439845, 0.07941484, 0.09641262, 0.10352695],
[0.0885492 , 0.07412052, 0.11018585, 0.12078145],
[0.09961785, 0.07941484, 0.11412106, 0.09202396],
[0.10238501, 0.07412052, 0.12002387, 0.10927845],
[0.10930291, 0.10059213, 0.12592669, 0.11502995],
[0.0885492 , 0.07412052, 0.11018585, 0.12653294],
```

```
       [0.08716562, 0.07412052, 0.10034783, 0.08627246],
       [0.08439845, 0.06882619, 0.11018585, 0.08052096],
       [0.10653575, 0.07941484, 0.12002387, 0.13228444],
       [0.08716562, 0.09000348, 0.11018585, 0.13803594],
       [0.0885492 , 0.082062  , 0.10821824, 0.10352695],
       [0.08301487, 0.07941484, 0.09444501, 0.10352695],
       [0.0954671 , 0.082062  , 0.10625064, 0.12078145],
       [0.09269994, 0.082062  , 0.11018585, 0.13803594],
       [0.0954671 , 0.082062  , 0.10034783, 0.13228444],
       [0.08024771, 0.07147336, 0.10034783, 0.10927845],
       [0.09408352, 0.08470916, 0.11608866, 0.13228444],
       [0.09269994, 0.08735632, 0.11215345, 0.14378743],
       [0.09269994, 0.07941484, 0.10231543, 0.13228444],
       [0.08716562, 0.06617903, 0.09838022, 0.10927845],
       [0.08993278, 0.07941484, 0.10231543, 0.11502995],
       [0.08578203, 0.09000348, 0.10625064, 0.13228444],
       [0.08163129, 0.07941484, 0.10034783, 0.10352695]])
```

```
# Split the dataset into training and testing sets
x_train,x_test,y_train,y_test =
train_test_split(x_nor,y,test_size=0.2,random_state=50)

print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

```
(120, 4) (120,)
(30, 4) (30,)
```

```
from keras.utils import np_utils
y_train1 = np_utils.to_categorical(y_train,num_classes=3)
y_test1 = np_utils.to_categorical(y_test,num_classes=3)

print("shape of y_train",y_train1.shape)
print("shape of y_test",y_test1.shape)
```

```
shape of y_train (120, 3)
shape of y_test (30, 3)
```

```
# Define the model architecture
model = Sequential()
model.add(Dense(32, activation = "relu", input_dim = 4))    # inpte
layer
model.add(Dense(64, activation = "relu"))                    #1st
hiden layer
model.add(Dense(3, activation = "softmax"))                   #output
layer

# Compile the model
model.compile(optimizer="adam",                     # optimizer adjest
the model wieghts to maximize a loss fuctions.
             loss = "categorical_crossentropy",
```

```
             metrics = ['accuracy'])                # For finalizing
the model & make it comletely ready for use

history = model.fit(x_train, y_train1, batch_size=20, epochs=47,
verbose=1, validation_data=(x_test, y_test1))
history

Epoch 1/47
6/6 [==============================] - 0s 22ms/step - loss: 0.1971 -
accuracy: 0.9833 - val_loss: 0.2189 - val_accuracy: 0.9333
Epoch 2/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1934 -
accuracy: 0.9833 - val_loss: 0.2051 - val_accuracy: 0.9667
Epoch 3/47
6/6 [==============================] - 0s 10ms/step - loss: 0.1912 -
accuracy: 0.9667 - val_loss: 0.1953 - val_accuracy: 0.9667
Epoch 4/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1852 -
accuracy: 0.9833 - val_loss: 0.2118 - val_accuracy: 0.9333
Epoch 5/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1831 -
accuracy: 0.9750 - val_loss: 0.1969 - val_accuracy: 0.9667
Epoch 6/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1788 -
accuracy: 0.9750 - val_loss: 0.1860 - val_accuracy: 0.9667
Epoch 7/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1749 -
accuracy: 0.9667 - val_loss: 0.1884 - val_accuracy: 0.9667
Epoch 8/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1719 -
accuracy: 0.9750 - val_loss: 0.2043 - val_accuracy: 0.9333
Epoch 9/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1688 -
accuracy: 0.9750 - val_loss: 0.1854 - val_accuracy: 0.9667
Epoch 10/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1653 -
accuracy: 0.9750 - val_loss: 0.1718 - val_accuracy: 0.9667
Epoch 11/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1632 -
accuracy: 0.9750 - val_loss: 0.1729 - val_accuracy: 0.9667
Epoch 12/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1593 -
accuracy: 0.9750 - val_loss: 0.1863 - val_accuracy: 0.9333
Epoch 13/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1579 -
accuracy: 0.9750 - val_loss: 0.1688 - val_accuracy: 0.9667
Epoch 14/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1549 -
accuracy: 0.9750 - val_loss: 0.1768 - val_accuracy: 0.9667
Epoch 15/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1510 -
```

```
accuracy: 0.9750 - val_loss: 0.1648 - val_accuracy: 0.9667
Epoch 16/47
6/6 [==============================] - 0s 10ms/step - loss: 0.1500 -
accuracy: 0.9750 - val_loss: 0.1736 - val_accuracy: 0.9667
Epoch 17/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1509 -
accuracy: 0.9500 - val_loss: 0.1499 - val_accuracy: 0.9667
Epoch 18/47
6/6 [==============================] - 0s 10ms/step - loss: 0.1431 -
accuracy: 0.9750 - val_loss: 0.1715 - val_accuracy: 0.9333
Epoch 19/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1424 -
accuracy: 0.9833 - val_loss: 0.1739 - val_accuracy: 0.9333
Epoch 20/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1398 -
accuracy: 0.9833 - val_loss: 0.1626 - val_accuracy: 0.9667
Epoch 21/47
6/6 [==============================] - 0s 10ms/step - loss: 0.1432 -
accuracy: 0.9500 - val_loss: 0.1422 - val_accuracy: 0.9667
Epoch 22/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1370 -
accuracy: 0.9750 - val_loss: 0.1696 - val_accuracy: 0.9333
Epoch 23/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1354 -
accuracy: 0.9833 - val_loss: 0.1533 - val_accuracy: 0.9667
Epoch 24/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1321 -
accuracy: 0.9750 - val_loss: 0.1620 - val_accuracy: 0.9333
Epoch 25/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1297 -
accuracy: 0.9833 - val_loss: 0.1480 - val_accuracy: 0.9667
Epoch 26/47
6/6 [==============================] - 0s 10ms/step - loss: 0.1295 -
accuracy: 0.9667 - val_loss: 0.1389 - val_accuracy: 0.9667
Epoch 27/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1265 -
accuracy: 0.9750 - val_loss: 0.1471 - val_accuracy: 0.9667
Epoch 28/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1267 -
accuracy: 0.9833 - val_loss: 0.1585 - val_accuracy: 0.9333
Epoch 29/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1239 -
accuracy: 0.9750 - val_loss: 0.1397 - val_accuracy: 0.9667
Epoch 30/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1221 -
accuracy: 0.9750 - val_loss: 0.1465 - val_accuracy: 0.9667
Epoch 31/47
6/6 [==============================] - 0s 10ms/step - loss: 0.1209 -
accuracy: 0.9750 - val_loss: 0.1360 - val_accuracy: 0.9667
Epoch 32/47
```

```
6/6 [==============================] - 0s 11ms/step - loss: 0.1185 -
accuracy: 0.9750 - val_loss: 0.1467 - val_accuracy: 0.9667
Epoch 33/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1184 -
accuracy: 0.9750 - val_loss: 0.1402 - val_accuracy: 0.9667
Epoch 34/47
6/6 [==============================] - 0s 13ms/step - loss: 0.1170 -
accuracy: 0.9667 - val_loss: 0.1361 - val_accuracy: 0.9667
Epoch 35/47
6/6 [==============================] - 0s 15ms/step - loss: 0.1175 -
accuracy: 0.9750 - val_loss: 0.1639 - val_accuracy: 0.9333
Epoch 36/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1139 -
accuracy: 0.9833 - val_loss: 0.1323 - val_accuracy: 0.9667
Epoch 37/47
6/6 [==============================] - 0s 14ms/step - loss: 0.1135 -
accuracy: 0.9667 - val_loss: 0.1225 - val_accuracy: 0.9667
Epoch 38/47
6/6 [==============================] - 0s 14ms/step - loss: 0.1128 -
accuracy: 0.9583 - val_loss: 0.1336 - val_accuracy: 0.9667
Epoch 39/47
6/6 [==============================] - 0s 13ms/step - loss: 0.1129 -
accuracy: 0.9833 - val_loss: 0.1640 - val_accuracy: 0.9333
Epoch 40/47
6/6 [==============================] - 0s 10ms/step - loss: 0.1126 -
accuracy: 0.9750 - val_loss: 0.1248 - val_accuracy: 0.9667
Epoch 41/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1087 -
accuracy: 0.9750 - val_loss: 0.1286 - val_accuracy: 0.9667
Epoch 42/47
6/6 [==============================] - 0s 10ms/step - loss: 0.1072 -
accuracy: 0.9667 - val_loss: 0.1299 - val_accuracy: 0.9667
Epoch 43/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1044 -
accuracy: 0.9750 - val_loss: 0.1486 - val_accuracy: 0.9333
Epoch 44/47
6/6 [==============================] - 0s 11ms/step - loss: 0.1062 -
accuracy: 0.9833 - val_loss: 0.1474 - val_accuracy: 0.9333
Epoch 45/47
6/6 [==============================] - 0s 10ms/step - loss: 0.1071 -
accuracy: 0.9750 - val_loss: 0.1158 - val_accuracy: 0.9667
Epoch 46/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1029 -
accuracy: 0.9667 - val_loss: 0.1289 - val_accuracy: 0.9667
Epoch 47/47
6/6 [==============================] - 0s 12ms/step - loss: 0.1011 -
accuracy: 0.9750 - val_loss: 0.1351 - val_accuracy: 0.9667

<keras.callbacks.History at 0x21fcb7e3370>
```

```
# Calculating F1-Score
y_train_pred = model.predict(x_train).round()
y_test_pred = model.predict(x_test).round()
```

```
4/4 [==============================] - 0s 4ms/step
1/1 [==============================] - 0s 31ms/step
```

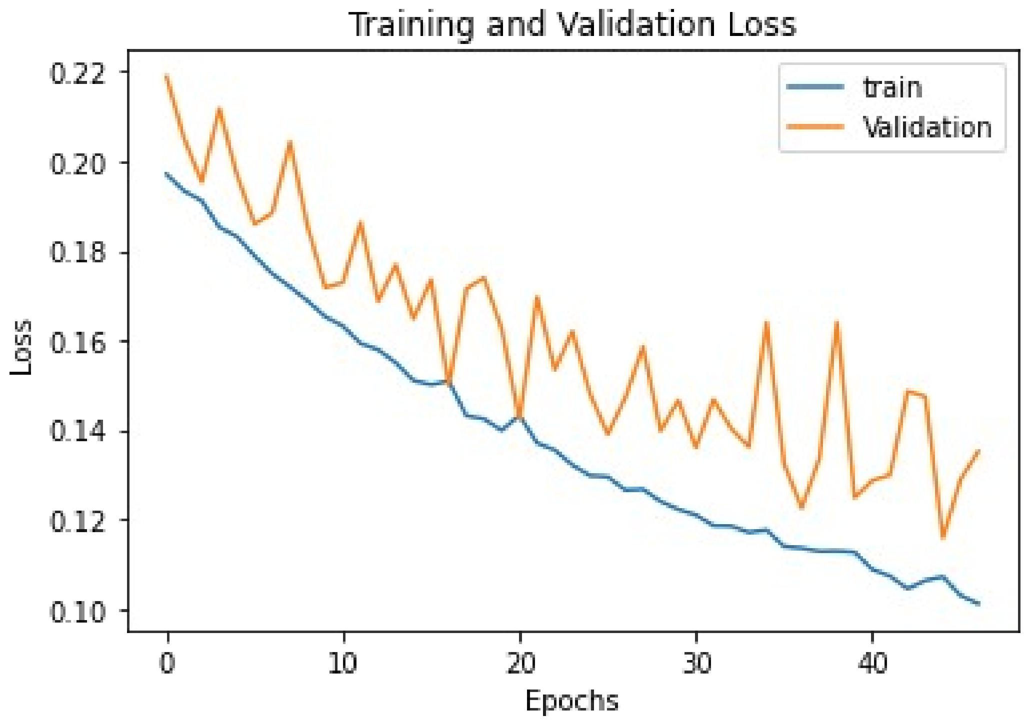
```
from sklearn.metrics import accuracy_score
test_accuracy = accuracy_score(y_test1, y_test_pred)
train_accuracy = accuracy_score(y_train1, y_train_pred)

# Print the evaluation metrics
print("Train Loss:", train_loss*100)
print("Test Loss:", test_loss*100)
print("Train Accuracy:", train_accuracy*100)
print("Test Accuracy:", test_accuracy*100)
```

```
Train Loss: 7.071732729673386
Test Loss: 5.355305224657059
Train Accuracy: 97.5
Test Accuracy: 96.66666666666667
```

```
# Ploting the loss function
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(["train", "Validation"], loc="upper right")
```

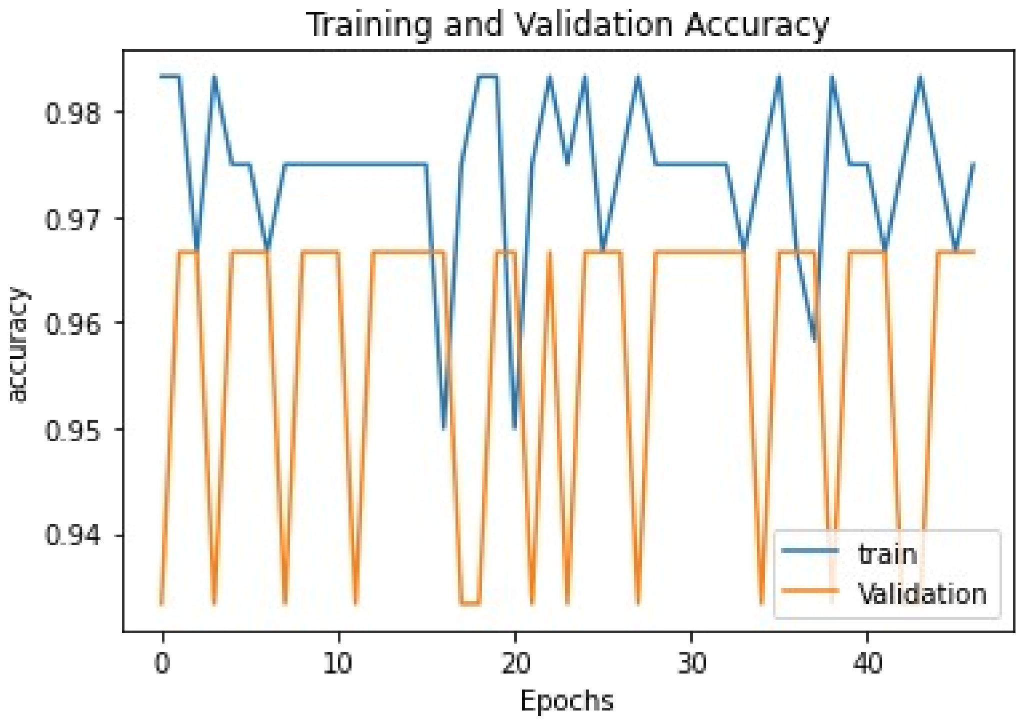
```
<matplotlib.legend.Legend at 0x21fcb780a90>
```




```
# Ploting the accuracy function
print(history.history.keys())
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('accuracy')
plt.legend(["train", "Validation"], loc="lower right")
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
<matplotlib.legend.Legend at 0x21fccf60700>
```




```
print("Train Accuracy:", train_accuracy*100)
print("Test Accuracy:", test_accuracy*100)
```

```
Train Accuracy: 97.5
Test Accuracy: 96.66666666666667
```