In [1]:
```python
#QUESTION i
#Preprocessing Phase
# Task1: Selection
#Select Necessary Libraries
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import Binarizer
from sklearn.decomposition import PCA
#Step2: Select Data Source
path='c:/pydatafiles/KisumuCountyFoodSecurity.csv'
Data_Source=pd.read_csv(path)
print("Selected Data Source:\n",Data_Source)
```

```
Selected Data Source:
       Subcounty  Population  Monthly consumption (90 Kg bags)  Currentstocks  \
0          Seme       98805                            1650.0           4750
1   Kisumu East      453890                            7575.0          29650
2        Nyando      141037                            2354.0           4960
3       Nyakach      144571                            2413.0           6820
4   Kisumu west       97698                            1631.0           4780
..          ...         ...                               ...            ...
67       Nyando      141037                             470.0           4510
68      Nyakach      144571                             482.0           3156
69  Kisumu west       97698                             326.0           2558
70     Muhoroni      137065                             457.0           2196
71         Seme       98805                             329.4           3556

    Produce
0     Beans
1     Beans
2     Beans
3     Beans
4     Beans
..      ...
67  Sorghum
68  Sorghum
69  Sorghum
70  Sorghum
71  Sorghum

[72 rows x 5 columns]
```

In [2]:
```python
#QUESTION i
dups = Data_Source.duplicated()
# report if there are any duplicates
print(dups.any())
# list all duplicate rows
print(Data_Source[dups])
```

```
True
       Subcounty  Population  Monthly consumption (90 Kg bags)  Currentstocks  \
55  Kisumu East      453890                           41295.0         100080
56       Nyando      141037                           12832.0          47500
57      Nyakach      144571                           13153.0          43750
58  Kisumu west       97698                            8889.0          46885
59     Muhoroni      137065                           12470.0          36680

   Produce
55   Maize
56   Maize
57   Maize
58   Maize
59   Maize
```

In [3]:
```python
#QUESTION i
null_counts = Data_Source.isnull().sum()
print("Number of null values in each column:\n{}".format(null_counts))
```

```
Number of null values in each column:
Subcounty                           0
Population                          0
Monthly consumption (90 Kg bags)    0
Currentstocks                       0
Produce                             0
dtype: int64
```

In [4]:
```python
#QUESTION i
#Step3: Data Selection(Sampling)
#Select the first records using Head() function
Sample1=Data_Source.head()
print("The first Records\n",Sample1)
Sample2=Data_Source.head(10)
print("\nThe first four records\n",Sample2)
```

```
The first Records
      Subcounty  Population  Monthly consumption (90 Kg bags)  Currentstocks  \
0         Seme       98805                            1650.0           4750
1  Kisumu East      453890                            7575.0          29650
2       Nyando      141037                            2354.0           4960
3      Nyakach      144571                            2413.0           6820
4  Kisumu west       97698                            1631.0           4780

  Produce
0   Beans
1   Beans
2   Beans
3   Beans
4   Beans

The first four records
      Subcounty  Population  Monthly consumption (90 Kg bags)  Currentstocks  \
0         Seme       98805                            1650.0           4750
1  Kisumu East      453890                            7575.0          29650
2       Nyando      141037                            2354.0           4960
3      Nyakach      144571                            2413.0           6820
4  Kisumu west       97698                            1631.0           4780
5     Muhoroni      137065                            2288.0           6480
6         Seme       98805                            8990.0          62560
7  Kisumu East      453890                           41295.0         100080
8       Nyando      141037                           12832.0          47500
9      Nyakach      144571                           13153.0          43750

  Produce
0   Beans
1   Beans
2   Beans
3   Beans
4   Beans
5   Beans
6   Maize
7   Maize
8   Maize
9   Maize
```

In [5]:
```python
#Question i
#Select the last n records using tail() function
Sample3=Data_Source.tail()
Sample4=Data_Source.tail(20)
print("\n The last three records\n",Sample4)
#Select a Random sample of size n using sample() function
Sample5=Data_Source.sample(20)
print("\nThe data random sample records\n",Sample5)
```

```
 The last three records
       Subcounty  Population  Monthly consumption (90 Kg bags)  Currentstocks  \
52  Kisumu west       97698                            1631.0           2875
53     Muhoroni      137065                            2288.0           3320
54         Seme       98805                            8990.0          44580
55  Kisumu East      453890                           41295.0         100080
56       Nyando      141037                           12832.0          47500
57      Nyakach      144571                           13153.0          43750
58  Kisumu west       97698                            8889.0          46885
59     Muhoroni      137065                           12470.0          36680
60         Seme       98805                            1412.0            750
61  Kisumu East      453890                            6484.0          24340
62       Nyando      141037                            2015.0          42250
63      Nyakach      144571                            2065.0          13145
64  Kisumu west       97698                            1600.0            910
65     Muhoroni      137065                            1958.0          37960
66  Kisumu East      453890                            1513.0           2984
67       Nyando      141037                             470.0           4510
68      Nyakach      144571                             482.0           3156
69  Kisumu west       97698                             326.0           2558
70     Muhoroni      137065                             457.0           2196
71         Seme       98805                             329.4           3556

    Produce
52    Beans
53    Beans
54    Maize
55    Maize
56    Maize
57    Maize
58    Maize
59    Maize
60     Rice
61     Rice
62     Rice
63     Rice
64     Rice
65     Rice
66  Sorghum
67  Sorghum
68  Sorghum
69  Sorghum
70  Sorghum
71  Sorghum

The data random sample records
       Subcounty  Population  Monthly consumption (90 Kg bags)  Currentstocks  \
43       Nyando      141037                             470.0           4980
25  Kisumu East      453890                            7575.0          22075
64  Kisumu west       97698                            1600.0            910
32       Nyando      141037                           12832.0          34668
68      Nyakach      144571                             482.0           3156
38       Nyando      141037                            2015.0          48850
0          Seme       98805                            1650.0           4750
59     Muhoroni      137065                           12470.0          36680
50       Nyando      141037                            2354.0           3460
16  Kisumu west       97698                            1600.0            550
26       Nyando      141037                            2354.0           2606
24         Seme       98805                            1650.0           3100
13  Kisumu East      453890                            6484.0          10112
58  Kisumu west       97698                            8889.0          46885
34  Kisumu west       97698                            8889.0          37996
51      Nyakach      144571                            2413.0           3845
71         Seme       98805                             329.4           3556
19       Nyando      141037                             470.0           5450
10  Kisumu west       97698                            8889.0          46885
21  Kisumu west       97698                             326.0           3210

    Produce
43  Sorghum
25    Beans
64     Rice
32    Maize
68  Sorghum
38     Rice
0     Beans
59    Maize
50    Beans
16     Rice
26    Beans
24    Beans
13     Rice
58    Maize
34    Maize
51    Beans
71  Sorghum
19  Sorghum
10    Maize
21  Sorghum
```

In [6]:
```python
#Question i
#Data Transformation
#i. labelling
#Transform the data into numerals/numbers
Transformed_data=Data_Source.apply(preprocessing.LabelEncoder().fit_transform)
print("Transformed Data\n",Transformed_data)
#Separate/extract numeric values/Array from the attributes
Numeric_values=Transformed_data.values
print("Numeric Values or Array\n",Numeric_values)
```

```
Transformed Data
     Subcounty  Population  Monthly consumption (90 Kg bags)  Currentstocks  \
0           8           1                                 9             31
1           0           5                                17             50
2           6           3                                14             33
3           4           4                                15             39
4           1           0                                 8             32
..        ...         ...                               ...            ...
67          7           3                                 3             30
68          5           4                                 4             17
69          1           0                                 0              7
70          3           2                                 2              6
71          8           1                                 1             22

    Produce
0         0
1         0
2         0
3         0
4         0
..      ...
67        3
68        3
69        3
70        3
71        3

[72 rows x 5 columns]
Numeric Values or Array
 [[ 8  1  9 31  0]
 [ 0  5 17 50  0]
 [ 6  3 14 33  0]
 [ 4  4 15 39  0]
 [ 1  0  8 32  0]
 [ 2  2 13 38  0]
 [ 8  1 19 65  1]
 [ 0  5 23 66  1]
 [ 6  3 21 61  1]
 [ 4  4 22 58  1]
 [ 1  0 18 60  1]
 [ 3  2 20 53  1]
 [ 8  1  5  0  2]
 [ 0  5 16 40  2]
 [ 7  3 11 41  2]
 [ 5  4 12 36  2]
 [ 1  0  7  1  2]
 [ 3  2 10 42  2]
 [ 0  5  6 35  3]
 [ 7  3  3 37  3]
 [ 5  4  4 26  3]
 [ 1  0  0 18  3]
 [ 3  2  2 15  3]
 [ 8  1  1 28  3]
 [ 8  1  9 14  0]
 [ 0  5 17 46  0]
 [ 6  3 14  8  0]
 [ 4  4 15 29  0]
 [ 1  0  8 16  0]
 [ 2  2 13 27  0]
 [ 8  1 19 63  1]
 [ 0  5 23 64  1]
 [ 6  3 21 52  1]
 [ 4  4 22 51  1]
 [ 1  0 18 55  1]
 [ 3  2 20 48  1]
 [ 8  1  5  2  2]
 [ 0  5 16 47  2]
 [ 7  3 11 62  2]
 [ 5  4 12 44  2]
 [ 1  0  7  4  2]
 [ 3  2 10 56  2]
 [ 0  5  6 21  3]
 [ 7  3  3 34  3]
 [ 5  4  4 23  3]
 [ 1  0  0 12  3]
 [ 3  2  2  9  3]
 [ 8  1  1 25  3]
 [ 8  1  9 10  0]
 [ 0  5 17 45  0]
 [ 6  3 14 20  0]
 [ 4  4 15 24  0]
 [ 1  0  8 11  0]
 [ 2  2 13 19  0]
 [ 8  1 19 59  1]
 [ 0  5 23 66  1]
 [ 6  3 21 61  1]
 [ 4  4 22 58  1]
 [ 1  0 18 60  1]
 [ 3  2 20 53  1]
 [ 8  1  5  3  2]
 [ 0  5 16 49  2]
 [ 7  3 11 57  2]
 [ 5  4 12 43  2]
 [ 1  0  7  5  2]
 [ 3  2 10 54  2]
 [ 0  5  6 13  3]
 [ 7  3  3 30  3]
 [ 5  4  4 17  3]
 [ 1  0  0  7  3]
 [ 3  2  2  6  3]
 [ 8  1  1 22  3]]
```

In [7]:
```python
#Question ii number 1
#TRAINING DATA
split=44
#Select Training Sample using slicer
Train_Sample=Numeric_values[:split,:]
print("\nTraining Sample\n",Train_Sample)
```

```
Training Sample
 [[ 8  1  9 31  0]
 [ 0  5 17 50  0]
 [ 6  3 14 33  0]
 [ 4  4 15 39  0]
 [ 1  0  8 32  0]
 [ 2  2 13 38  0]
 [ 8  1 19 65  1]
 [ 0  5 23 66  1]
 [ 6  3 21 61  1]
 [ 4  4 22 58  1]
 [ 1  0 18 60  1]
 [ 3  2 20 53  1]
 [ 8  1  5  0  2]
 [ 0  5 16 40  2]
 [ 7  3 11 41  2]
 [ 5  4 12 36  2]
 [ 1  0  7  1  2]
 [ 3  2 10 42  2]
 [ 0  5  6 35  3]
 [ 7  3  3 37  3]
 [ 5  4  4 26  3]
 [ 1  0  0 18  3]
 [ 3  2  2 15  3]
 [ 8  1  1 28  3]
 [ 8  1  9 14  0]
 [ 0  5 17 46  0]
 [ 6  3 14  8  0]
 [ 4  4 15 29  0]
 [ 1  0  8 16  0]
 [ 2  2 13 27  0]
 [ 8  1 19 63  1]
 [ 0  5 23 64  1]
 [ 6  3 21 52  1]
 [ 4  4 22 51  1]
 [ 1  0 18 55  1]
 [ 3  2 20 48  1]
 [ 8  1  5  2  2]
 [ 0  5 16 47  2]
 [ 7  3 11 62  2]
 [ 5  4 12 44  2]
 [ 1  0  7  4  2]
 [ 3  2 10 56  2]
 [ 0  5  6 21  3]
 [ 7  3  3 34  3]]
```

In [8]:
```python
#Question ii number 2
split=44
#Select Test Sample using slicer
Test_Sample=Numeric_values[split:,:]
print("\nTest Sample\n",Test_Sample)
```

```
Test Sample
 [[ 5  4  4 23  3]
 [ 1  0  0 12  3]
 [ 3  2  2  9  3]
 [ 8  1  1 25  3]
 [ 8  1  9 10  0]
 [ 0  5 17 45  0]
 [ 6  3 14 20  0]
 [ 4  4 15 24  0]
 [ 1  0  8 11  0]
 [ 2  2 13 19  0]
 [ 8  1 19 59  1]
 [ 0  5 23 66  1]
 [ 6  3 21 61  1]
 [ 4  4 22 58  1]
 [ 1  0 18 60  1]
 [ 3  2 20 53  1]
 [ 8  1  5  3  2]
 [ 0  5 16 49  2]
 [ 7  3 11 57  2]
 [ 5  4 12 43  2]
 [ 1  0  7  5  2]
 [ 3  2 10 54  2]
 [ 0  5  6 13  3]
 [ 7  3  3 30  3]
 [ 5  4  4 17  3]
 [ 1  0  0  7  3]
 [ 3  2  2  6  3]
 [ 8  1  1 22  3]]
```

In [9]:
```python
#question ii number 3
#Select input attributes sample and output attribute sample
#Select input attributes from numeric Training Sample
Train_X_Sample=Train_Sample[:,:-1]
print("\n Input attribute Sample from Train Sample\n",Train_X_Sample)
#b.Select output Attributes from numeric Training Sample
```

```
 Input attribute Sample from Train Sample
 [[ 8  1  9 31]
 [ 0  5 17 50]
 [ 6  3 14 33]
 [ 4  4 15 39]
 [ 1  0  8 32]
 [ 2  2 13 38]
 [ 8  1 19 65]
 [ 0  5 23 66]
 [ 6  3 21 61]
 [ 4  4 22 58]
 [ 1  0 18 60]
 [ 3  2 20 53]
 [ 8  1  5  0]
 [ 0  5 16 40]
 [ 7  3 11 41]
 [ 5  4 12 36]
 [ 1  0  7  1]
 [ 3  2 10 42]
 [ 0  5  6 35]
 [ 7  3  3 37]
 [ 5  4  4 26]
 [ 1  0  0 18]
 [ 3  2  2 15]
 [ 8  1  1 28]
 [ 8  1  9 14]
 [ 0  5 17 46]
 [ 6  3 14  8]
 [ 4  4 15 29]
 [ 1  0  8 16]
 [ 2  2 13 27]
 [ 8  1 19 63]
 [ 0  5 23 64]
 [ 6  3 21 52]
 [ 4  4 22 51]
 [ 1  0 18 55]
 [ 3  2 20 48]
 [ 8  1  5  2]
 [ 0  5 16 47]
 [ 7  3 11 62]
 [ 5  4 12 44]
 [ 1  0  7  4]
 [ 3  2 10 56]
 [ 0  5  6 21]
 [ 7  3  3 34]]
```

In [10]:
```python
Train_Y_Sample=Train_Sample[:,-1]
print("\n output attribute from Training Sample\n",Train_Y_Sample)
```

```
 output attribute from Training Sample
 [0 0 0 0 0 0 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 0 0 0 0 0 0 1 1 1 1 1 1 2
 2 2 2 2 2 3 3]
```

In [11]:
```python
#question ii number 4
#Select input attributes from numeric Training Sample
Test_X_Sample=Test_Sample[:,:-1]
print("\n Input attribute Sample from Train Sample\n",Train_X_Sample)
#b.Select output Attributes from numeric Training Sample
```

```
 Input attribute Sample from Train Sample
 [[ 8  1  9 31]
 [ 0  5 17 50]
 [ 6  3 14 33]
 [ 4  4 15 39]
 [ 1  0  8 32]
 [ 2  2 13 38]
 [ 8  1 19 65]
 [ 0  5 23 66]
 [ 6  3 21 61]
 [ 4  4 22 58]
 [ 1  0 18 60]
 [ 3  2 20 53]
 [ 8  1  5  0]
 [ 0  5 16 40]
 [ 7  3 11 41]
 [ 5  4 12 36]
 [ 1  0  7  1]
 [ 3  2 10 42]
 [ 0  5  6 35]
 [ 7  3  3 37]
 [ 5  4  4 26]
 [ 1  0  0 18]
 [ 3  2  2 15]
 [ 8  1  1 28]
 [ 8  1  9 14]
 [ 0  5 17 46]
 [ 6  3 14  8]
 [ 4  4 15 29]
 [ 1  0  8 16]
 [ 2  2 13 27]
 [ 8  1 19 63]
 [ 0  5 23 64]
 [ 6  3 21 52]
 [ 4  4 22 51]
 [ 1  0 18 55]
 [ 3  2 20 48]
 [ 8  1  5  2]
 [ 0  5 16 47]
 [ 7  3 11 62]
 [ 5  4 12 44]
 [ 1  0  7  4]
 [ 3  2 10 56]
 [ 0  5  6 21]
 [ 7  3  3 34]]
```

In [12]:
```python
Test_Y_Sample=Test_Sample[:,-1]
print("\n output attribute from Training Sample\n",Train_Y_Sample)
```

```
 output attribute from Training Sample
 [0 0 0 0 0 0 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 0 0 0 0 0 0 1 1 1 1 1 1 2
 2 2 2 2 2 3 3]
```

In [13]:
```python
# scipy
import scipy
print('scipy: %s' % scipy.__version__)
# numpy
import numpy
print('numpy: %s' % numpy.__version__)
# matplotlib
import matplotlib
print('matplotlib: %s' % matplotlib.__version__)
# pandas
import pandas
print('pandas: %s' % pandas.__version__)
# statsmodels
import statsmodels
print('statsmodels: %s' % statsmodels.__version__)
# scikit-learn
import sklearn
print('sklearn: %s' % sklearn.__version__)
```

```
scipy: 1.6.1
numpy: 1.19.5
matplotlib: 3.3.4
pandas: 1.2.3
statsmodels: 0.12.2
sklearn: 0.24.1
```

In [14]:
```python
# theano
import theano
print('theano: %s' % theano.__version__)
# tensorflow
import tensorflow
print('tensorflow: %s' % tensorflow.__version__)
# keras
import keras
print('keras: %s' % keras.__version__)
```

```
theano: 1.0.4
tensorflow: 2.3.0
keras: 2.4.3
```

In [15]:
```python
# phase2: Preprocesing the Data
#Selection
#libraries to be used
from numpy import loadtxt
from keras.models import Sequential
from keras.layers import Dense
from keras.models import model_from_json
from ann_visualizer.visualize import ann_viz;
import pandas as pd
import numpy
import pydotplus
import os
os.environ["PATH"] +=os.pathsep + 'C:\Graphviz\bin'
from keras.utils.vis_utils import plot_model
```

In [16]:
```python
#phase 3: Model Creation

# define the keras model
model = Sequential()
model.add(Dense(12, input_dim=4, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
print(model.summary())
```

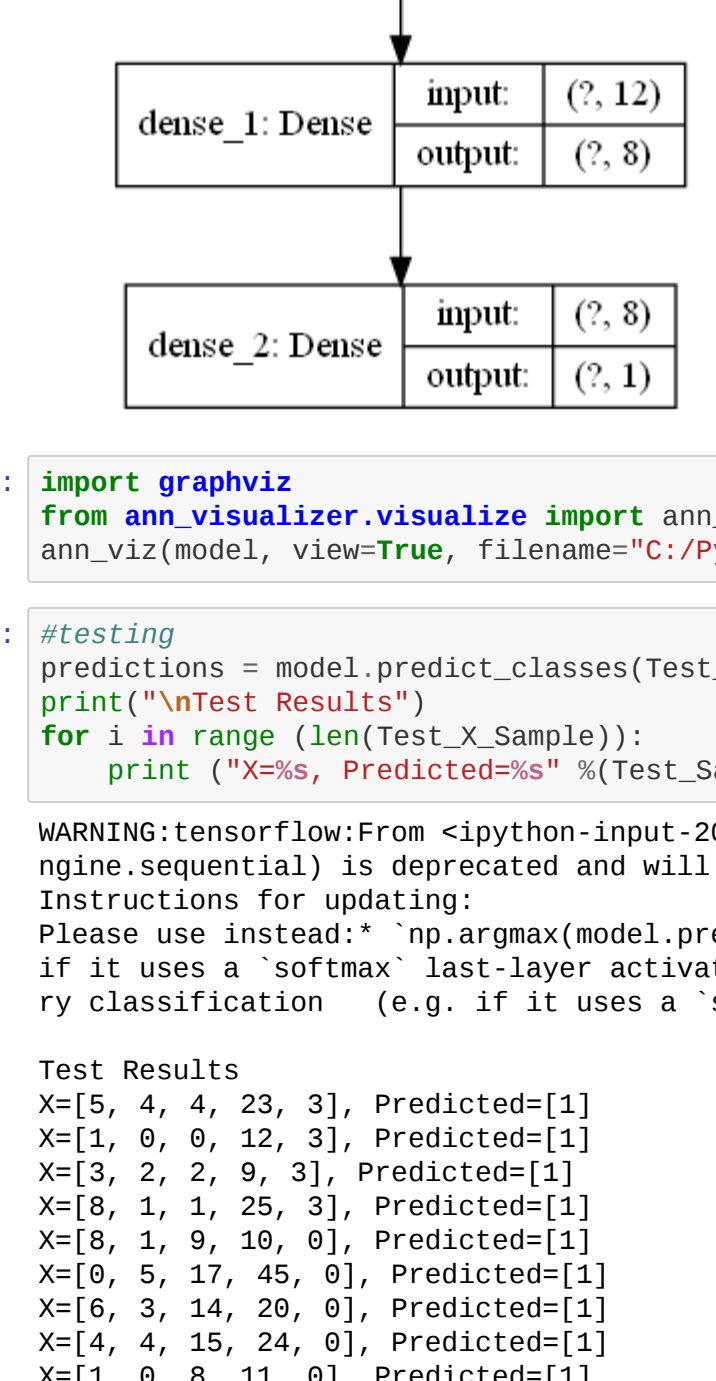
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 12)                60
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 104
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 9
=================================================================
Total params: 173
Trainable params: 173
Non-trainable params: 0
_________________________________________________________________
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 12)                60
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 104
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 9
=================================================================
Total params: 173
Trainable params: 173
Non-trainable params: 0
_________________________________________________________________
None
```

In [17]:
```python
#compile or execute the model
Compiled_Model=model.compile(loss='binary_crossentropy', optimizer='adam',metrics=['accuracy'])
print("Training Results")
Trained_model=model.fit(Train_X_Sample, Train_Y_Sample, epochs=50, batch_size=10)
```

```
Training Results
Epoch 1/50
5/5 [==============================] - 0s 2ms/step - loss: -0.0953 - accuracy: 0.3182
Epoch 2/50
5/5 [==============================] - 0s 2ms/step - loss: -0.4501 - accuracy: 0.2727
Epoch 3/50
5/5 [==============================] - 0s 1ms/step - loss: -0.8221 - accuracy: 0.2727
Epoch 4/50
5/5 [==============================] - 0s 4ms/step - loss: -1.1039 - accuracy: 0.2727
Epoch 5/50
5/5 [==============================] - 0s 3ms/step - loss: -1.3288 - accuracy: 0.2727
Epoch 6/50
5/5 [==============================] - 0s 4ms/step - loss: -1.5254 - accuracy: 0.2727
Epoch 7/50
5/5 [==============================] - 0s 2ms/step - loss: -1.7196 - accuracy: 0.2727
Epoch 8/50
5/5 [==============================] - 0s 4ms/step - loss: -1.8658 - accuracy: 0.2727
Epoch 9/50
5/5 [==============================] - 0s 3ms/step - loss: -1.9717 - accuracy: 0.2727
Epoch 10/50
5/5 [==============================] - 0s 2ms/step - loss: -2.1223 - accuracy: 0.2727
Epoch 11/50
5/5 [==============================] - 0s 4ms/step - loss: -2.2449 - accuracy: 0.2727
Epoch 12/50
5/5 [==============================] - 0s 2ms/step - loss: -2.3439 - accuracy: 0.2727
Epoch 13/50
5/5 [==============================] - 0s 2ms/step - loss: -2.4958 - accuracy: 0.2727
Epoch 14/50
5/5 [==============================] - 0s 3ms/step - loss: -2.6225 - accuracy: 0.2727
Epoch 15/50
5/5 [==============================] - 0s 2ms/step - loss: -2.7426 - accuracy: 0.2727
Epoch 16/50
5/5 [==============================] - 0s 3ms/step - loss: -2.8797 - accuracy: 0.2727
Epoch 17/50
5/5 [==============================] - 0s 4ms/step - loss: -3.0246 - accuracy: 0.2727
Epoch 18/50
5/5 [==============================] - 0s 4ms/step - loss: -3.1488 - accuracy: 0.2727
Epoch 19/50
5/5 [==============================] - 0s 3ms/step - loss: -3.3556 - accuracy: 0.2727
Epoch 20/50
5/5 [==============================] - 0s 3ms/step - loss: -3.4768 - accuracy: 0.2727
Epoch 21/50
5/5 [==============================] - 0s 3ms/step - loss: -3.6585 - accuracy: 0.2727
Epoch 22/50
5/5 [==============================] - 0s 3ms/step - loss: -3.7988 - accuracy: 0.2727
Epoch 23/50
5/5 [==============================] - 0s 3ms/step - loss: -4.0065 - accuracy: 0.2727
Epoch 24/50
5/5 [==============================] - 0s 3ms/step - loss: -4.2281 - accuracy: 0.2727
Epoch 25/50
5/5 [==============================] - ETA: 0s - loss: -7.3925 - accuracy: 0.100 - 0s 4ms/step - loss: -4.3926 - accu
racy: 0.2727
Epoch 26/50
5/5 [==============================] - 0s 4ms/step - loss: -4.6489 - accuracy: 0.2727
Epoch 27/50
5/5 [==============================] - 0s 4ms/step - loss: -4.8681 - accuracy: 0.2727
Epoch 28/50
5/5 [==============================] - 0s 4ms/step - loss: -5.1419 - accuracy: 0.2727
Epoch 29/50
5/5 [==============================] - 0s 4ms/step - loss: -5.4335 - accuracy: 0.2727
Epoch 30/50
5/5 [==============================] - 0s 3ms/step - loss: -5.7495 - accuracy: 0.2727
Epoch 31/50
5/5 [==============================] - 0s 2ms/step - loss: -6.1255 - accuracy: 0.2727
Epoch 32/50
5/5 [==============================] - 0s 4ms/step - loss: -6.6073 - accuracy: 0.2727
Epoch 33/50
5/5 [==============================] - 0s 2ms/step - loss: -7.0619 - accuracy: 0.2727
Epoch 34/50
5/5 [==============================] - 0s 3ms/step - loss: -7.4508 - accuracy: 0.2727
Epoch 35/50
5/5 [==============================] - 0s 3ms/step - loss: -7.9891 - accuracy: 0.2727
Epoch 36/50
5/5 [==============================] - 0s 3ms/step - loss: -8.4497 - accuracy: 0.2727
Epoch 37/50
5/5 [==============================] - 0s 2ms/step - loss: -8.9329 - accuracy: 0.2727
Epoch 38/50
5/5 [==============================] - 0s 4ms/step - loss: -9.4061 - accuracy: 0.2727
Epoch 39/50
5/5 [==============================] - 0s 2ms/step - loss: -9.9019 - accuracy: 0.2727
Epoch 40/50
5/5 [==============================] - 0s 3ms/step - loss: -10.3977 - accuracy: 0.2727
Epoch 41/50
5/5 [==============================] - 0s 2ms/step - loss: -10.8655 - accuracy: 0.2727
Epoch 42/50
5/5 [==============================] - 0s 2ms/step - loss: -11.4057 - accuracy: 0.2727
Epoch 43/50
5/5 [==============================] - 0s 2ms/step - loss: -11.8565 - accuracy: 0.2727
Epoch 44/50
5/5 [==============================] - 0s 3ms/step - loss: -12.4294 - accuracy: 0.2727
Epoch 45/50
5/5 [==============================] - 0s 2ms/step - loss: -12.9782 - accuracy: 0.2727
Epoch 46/50
5/5 [==============================] - 0s 3ms/step - loss: -13.6776 - accuracy: 0.2727
Epoch 47/50
5/5 [==============================] - 0s 3ms/step - loss: -14.3272 - accuracy: 0.2727
Epoch 48/50
5/5 [==============================] - 0s 2ms/step - loss: -15.0544 - accuracy: 0.2727
Epoch 49/50
5/5 [==============================] - 0s 3ms/step - loss: -15.9455 - accuracy: 0.2727
Epoch 50/50
5/5 [==============================] - 0s 3ms/step - loss: -16.4558 - accuracy: 0.2727
```

In [18]:
```python
import pydotplus
import pydot
import graphviz
import os
os.environ["PATH"] +=os.pathsep + 'C:/Graphviz/bin'
from keras.utils.vis_utils import plot_model
#visualize model
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

Out[18]:



In [19]:
```python
import graphviz
from ann_visualizer.visualize import ann_viz;
ann_viz(model, view=True, filename="C:/Pydatafiles/networksS4.gv", title="MyNeural Network")
```

In [20]:
```python
#testing
predictions = model.predict_classes(Test_X_Sample)
print("\nTest Results")
for i in range (len(Test_X_Sample)):
    print ("X=%s, Predicted=%s" %(Test_Sample[i].tolist(),predictions[i]))
```

```
WARNING:tensorflow:From <ipython-input-20-93f0d85f15>:2: Sequential.predict_classes (from tensorflow.python.keras.e
ngine.sequential) is deprecated and will be removed after 2021-01-01.
Instructions for updating:
Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g.
if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does bina
ry classification   (e.g. if it uses a `sigmoid` last-layer activation).

Test Results
X=[5, 4, 4, 23, 3], Predicted=[1]
X=[1, 0, 0, 12, 3], Predicted=[1]
X=[3, 2, 2, 9, 3], Predicted=[1]
X=[8, 1, 1, 25, 3], Predicted=[1]
X=[8, 1, 9, 10, 0], Predicted=[1]
X=[0, 5, 17, 45, 0], Predicted=[1]
X=[6, 3, 14, 20, 0], Predicted=[1]
X=[4, 4, 15, 24, 0], Predicted=[1]
X=[1, 0, 8, 11, 0], Predicted=[1]
X=[2, 2, 13, 19, 0], Predicted=[1]
X=[8, 1, 19, 59, 1], Predicted=[1]
X=[0, 5, 23, 66, 1], Predicted=[1]
X=[6, 3, 21, 61, 1], Predicted=[1]
X=[4, 4, 22, 58, 1], Predicted=[1]
X=[1, 0, 18, 60, 1], Predicted=[1]
X=[3, 2, 20, 53, 1], Predicted=[1]
X=[8, 1, 5, 3, 2], Predicted=[1]
X=[0, 5, 16, 49, 2], Predicted=[1]
X=[7, 3, 11, 57, 2], Predicted=[1]
X=[5, 4, 12, 43, 2], Predicted=[1]
X=[1, 0, 7, 5, 2], Predicted=[1]
X=[3, 2, 10, 54, 2], Predicted=[1]
X=[0, 5, 6, 13, 3], Predicted=[1]
X=[7, 3, 3, 30, 3], Predicted=[1]
X=[5, 4, 4, 17, 3], Predicted=[1]
X=[1, 0, 0, 7, 3], Predicted=[1]
X=[3, 2, 2, 6, 3], Predicted=[1]
X=[8, 1, 1, 22, 3], Predicted=[1]
```

In [21]:
```python
#Validate
_,accuracy = model.evaluate(Test_X_Sample, Test_Y_Sample)
print('Accuracy: %.2f' % (accuracy*100))
```

```
1/1 [==============================] - 0s 3ms/step - loss: -24.5349 - accuracy: 0.2143
Accuracy: 21.43
```

In [23]:
```python
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
#Create a Gaussian Classifier
gnb = GaussianNB()
gnb.fit(Train_X_Sample, Train_Y_Sample)
y_pred = gnb.predict(Test_X_Sample)
# Model Accuracy, how often is the classifier correct?
accuracy = metrics.accuracy_score(Test_Y_Sample, y_pred)
print("Accuracy: %.2f" % (accuracy*100))
```

```
Accuracy: 85.71
```