Car Evaluation

Shivam Porwal(1611CS14)

Debanjan Sarkar(1611CS17)

Background and Motivation

- We all Like Cars.
- Car Sell:
  - Around 3,50,000 cars are sold in India every year.
  - 5,15,000 cars are sold in Taiwan, 2005(which was highest in 10 years).

Introduction of Dataset

- Car Evaluation Database:
  - UCI Machine Learning Depository
  - https://archive.ics.uci.edu/ml/datasets/Car+Evaluation.
- Model Evaluates the cars based on certain characteristics.
- Derived from a Simple Hierarchical decision model.
- It is a Multiclass Classification Problem.

How Much Data is Enough ?..

How Much Data is Enough ?...

- We try to Discover the useful knowledge from volumes of data.

How Much Data is Enough ?...

- We try to Discover the useful knowledge from volumes of data.
- It Means larger data we use, better result we can get.

How Much Data is Enough ?...

- We try to Discover the useful knowledge from volumes of data.
- It Means larger data we use, better result we can get.
- But some scholar think that great deal of data don't guarantee better result than little of data.

How Much Data is Enough ?...

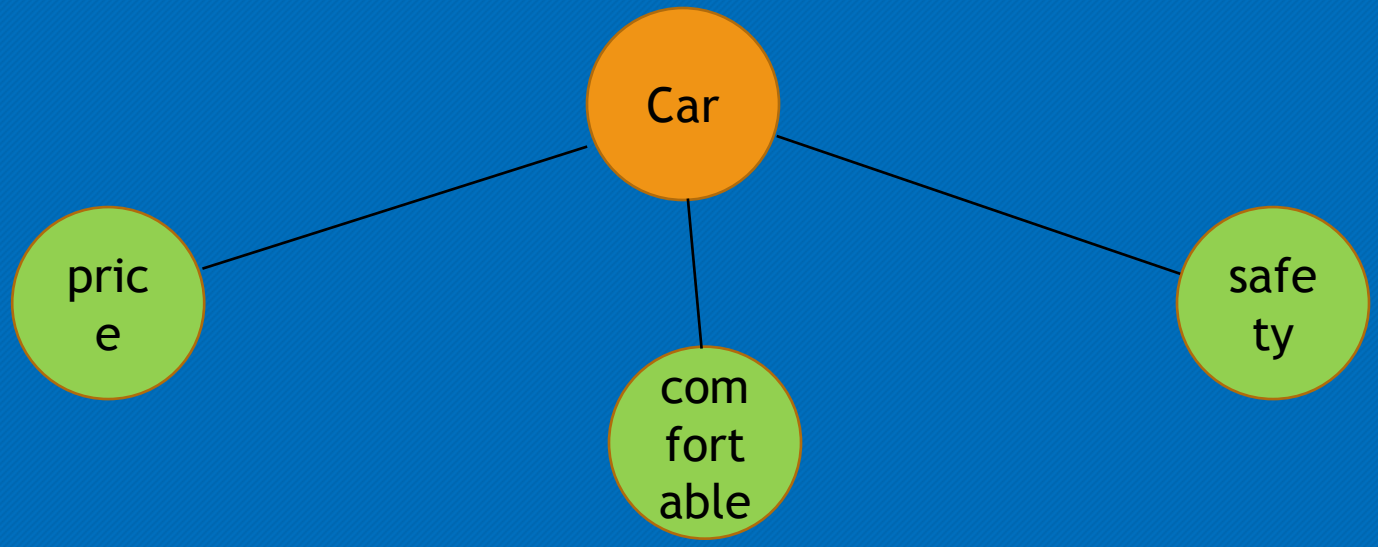
- We try to Discover the useful knowledge from volumes of data.
- It Means larger data we use, better result we can get.
- But some scholar think that great deal of data don't guarantee better result than little of data.
- Because resources are limited, the large samples will result in much load and contain lots of exceptional cases.

DataSet Information

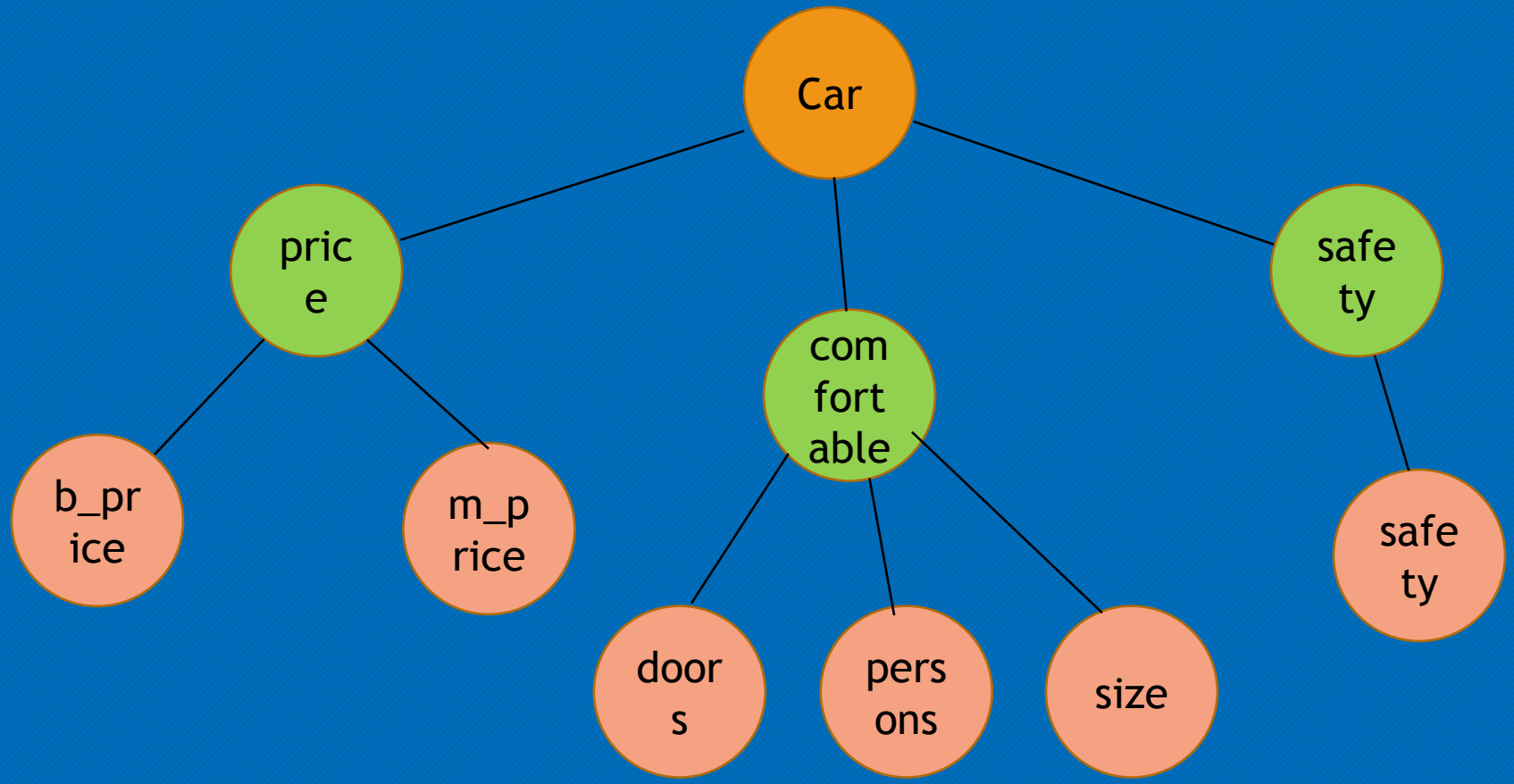
- Dataset consist of 1728 instances.
- Each record consist of 6 attributes:
  - Buying {very high, high, med, low}
  - MAINT {very high, high, med, low}
  - Doors {2,3,4,5-more}
  - Person {2,4,more}
  - LUG_BOOT {small, med, big}
  - Safety {low, Med, High}
- Class {unacc, acc, good, very-good}

Know the data

Know the data



Know the data



Approach

- Solution with neural network.
- As we cannot work with strings in neural networks, so we map the attribute values and classes to binary numbers.

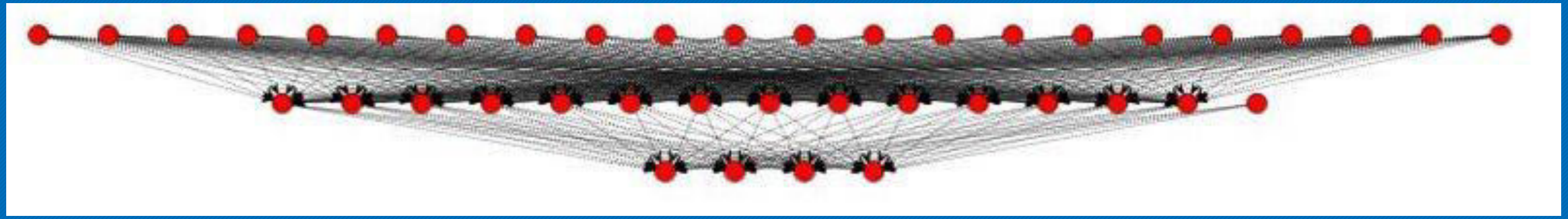
Transformed Dataset

buying: 1,0,0,0 instead of vhigh, 0,1,0,0 instead of high, 0,0,1,0 instead of med, 0,0,0,1 instead of low.
maint: 1,0,0,0 instead of vhigh, 0,1,0,0 instead of high, 0,0,1,0 instead of med, 0,0,0,1 instead of low.
doors: 0,0,0,1 instead of 2, 0,0,1,0 instead of 3, 0,1,0,0 instead of 4, 1,0,0,0 instead of 5more.
persons: 0,0,1 instead of 2, 0,1,0 instead of 4, 1,0,0 instead of more.
lug_boot: 0,0,1 instead of small, 0,1,0 instead of med, 1,0,0 instead of big.
safety: 0,0,1 instead of low, 0,1,0 instead of med, 1,0,0 instead of high.

Implementation

- Language used : Python
- Libraries used : Numpy, Keras
- Backend : Theano
- Type of Model : MLP(Multi-Layer Perceptron)

Network Architecture



Considerations:

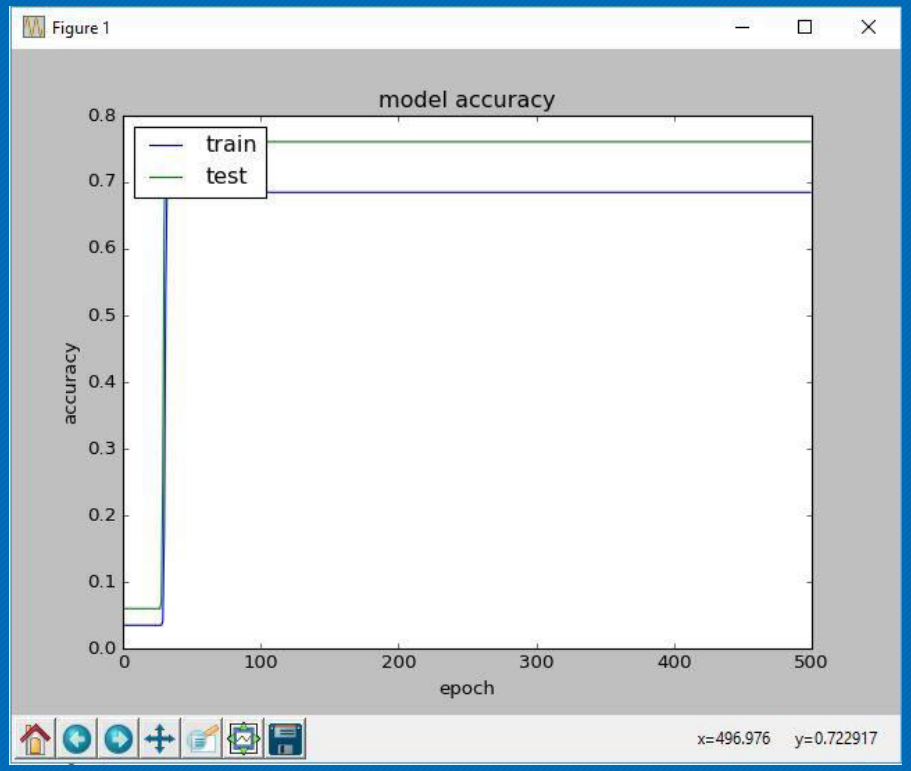
- CASE : 1
  - Input Neuron : 21
  - Number of Hidden layers:1
  - Hidden Neurons:14
  - Output Neurons : 4
- Sigmoid function is used as activation function in both Hidden Layer as well as output layer.
- Learning Rate is considered to be : 0.1.
- Momentum :0.7
- Number of Epochs: 500

Results:

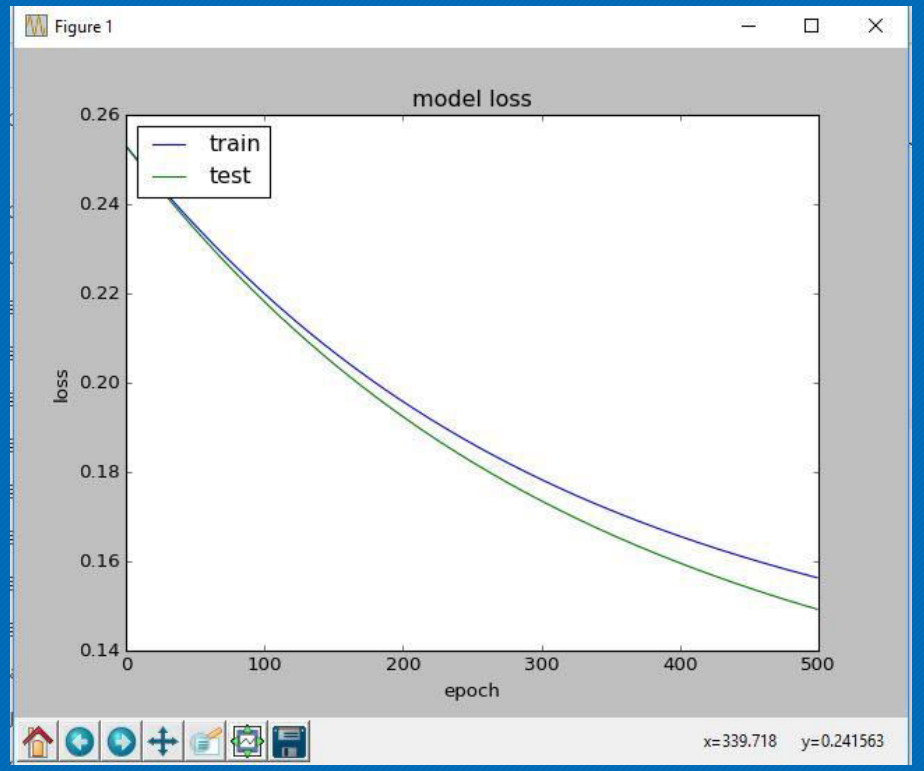
```
Epoch 497/500
1377/1377 [==============================] - 0s - loss: 0.1566 - acc: 0.6848 - val_loss: 0.1495 - val_acc: 0.7607
Epoch 498/500
1377/1377 [==============================] - 0s - loss: 0.1565 - acc: 0.6848 - val_loss: 0.1494 - val_acc: 0.7607
Epoch 499/500
1377/1377 [==============================] - 0s - loss: 0.1564 - acc: 0.6848 - val_loss: 0.1494 - val_acc: 0.7607
Epoch 500/500
1377/1377 [==============================] - 0s - loss: 0.1563 - acc: 0.6848 - val_loss: 0.1493 - val_acc: 0.7607
('mean squared error :', 0.14926735564344629)
('PREDICTED', array([[ 0.58279097,  0.37154564,  0.29080477,  0.28780967],
       [ 0.5833267 ,  0.37154239,  0.29027137,  0.28745881],
       [ 0.5828824 ,  0.37183481,  0.29136008,  0.28830126],
       ...,
       [ 0.58359236,  0.37079629,  0.29020908,  0.28789011],
       [ 0.58321053,  0.37072924,  0.29023898,  0.28795239],
       [ 0.58374566,  0.37072596,  0.28970632,  0.28760129]], dtype=float32))
('ORIGINAL', array([[1, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 0, 0, 0],
       ...,
       [1, 0, 0, 0],
       [0, 0, 1, 0],
       [0, 0, 0, 1]]))
```

Fig(a) : Output with Accuracy and MSE

Graphical Representation :



Fig(b) : Accuracy Plot



Fig(c) : Error Plot

Considerations:

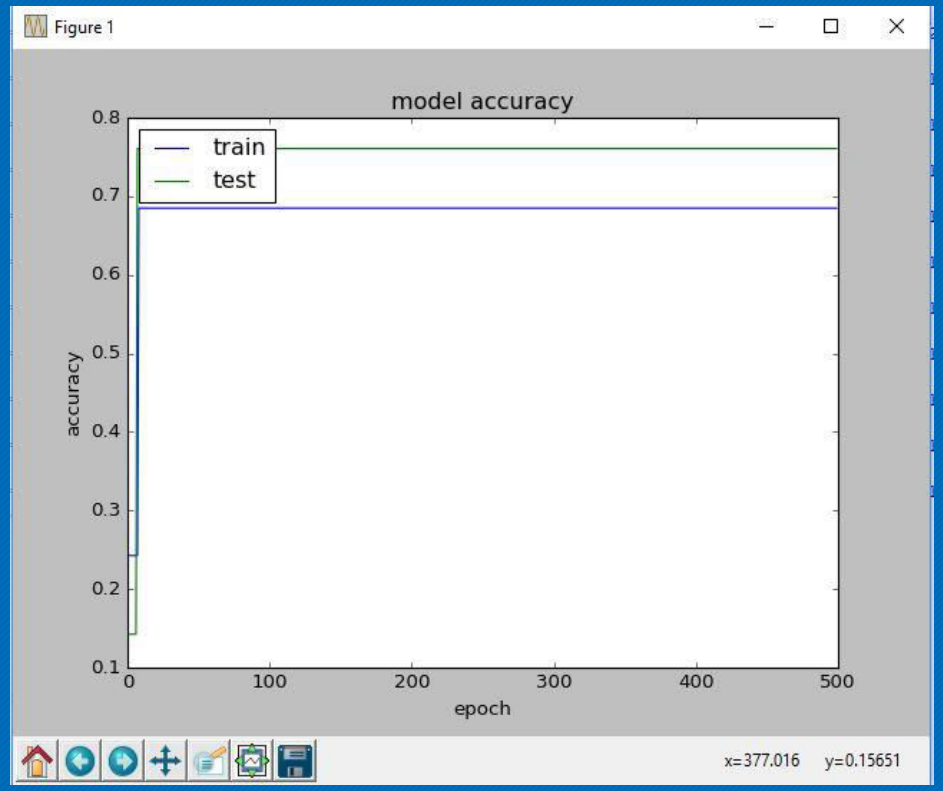
- CASE : 2
  - Input Neuron : 21
  - Number of Hidden layers:2
  - Neurons at hidden Layer 1: 14
  - Neurons at hidden Layer 2: 7
  - Output Neurons : 4
- Sigmoid function is used as activation function in both Hidden Layer as well as output layer.
- Learning Rate is considered to be : 0.1.
- Momentum :0.7
- Number of Epochs: 500

Results:

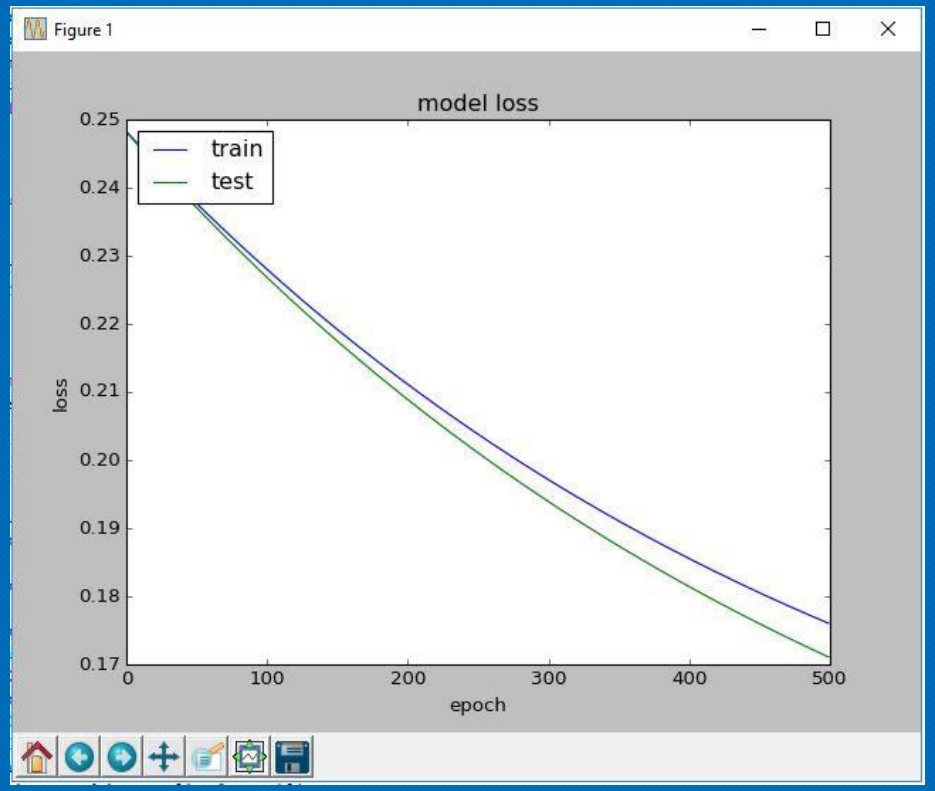
```
Epoch 497/500
1377/1377 [==============================] - 1s - loss: 0.1763 - acc: 0.6848 - val_loss: 0.1714 - val_acc: 0.7607
Epoch 498/500
1377/1377 [==============================] - 1s - loss: 0.1762 - acc: 0.6848 - val_loss: 0.1713 - val_acc: 0.7607
Epoch 499/500
1377/1377 [==============================] - 1s - loss: 0.1761 - acc: 0.6848 - val_loss: 0.1712 - val_acc: 0.7607
Epoch 500/500
1377/1377 [==============================] - 1s - loss: 0.1760 - acc: 0.6848 - val_loss: 0.1711 - val_acc: 0.7607
('mean squared error :', 0.17107827783140361)
('PREDICTED', array([[ 0.56394356,  0.41390133,  0.3456955 ,  0.34379134],
       [ 0.56394118,  0.41389966,  0.34568802,  0.3437905 ],
       [ 0.56394327,  0.4139027 ,  0.34569395,  0.34379551],
       ...,
       [ 0.56394738,  0.41388243,  0.34565979,  0.34375069],
       [ 0.56395102,  0.41387793,  0.34565696,  0.34374544],
       [ 0.56394863,  0.4138763 ,  0.34564945,  0.34374461]], dtype=float32))
('ORIGINAL', array([[1, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 0, 0, 0],
       ...,
       [1, 0, 0, 0],
       [0, 0, 1, 0],
       [0, 0, 0, 1]]))
```

Fig(d) : Output with Accuracy and MSE

Graphical Representation :



Fig(e) : Accuracy Plot



Fig(f) : Error Plot

Considerations:

- CASE : 3
  - Input Neuron : 21
  - Number of Hidden layers:3
  - Neurons at hidden Layer 1:500
  - Neurons at hidden Layer 2: 400
  - Neurons at hidden Layer 3: 300
  - Output Neurons : 4
- Sigmoid function is used as activation function in both Hidden Layer as well as output layer.
- Learning Rate is considered to be : 0.1.
- Momentum :0.7
- Number of Epochs: 20

Results:

```
1377/1377 [==============================] - 37s - loss: 0.2230 - acc: 0.6848 - val_loss: 0.2124 - val_acc: 0.7607
Epoch 10/20
1377/1377 [==============================] - 38s - loss: 0.2162 - acc: 0.6848 - val_loss: 0.2058 - val_acc: 0.7607
Epoch 11/20
1377/1377 [==============================] - 37s - loss: 0.2099 - acc: 0.6848 - val_loss: 0.1996 - val_acc: 0.7607
Epoch 12/20
1377/1377 [==============================] - 36s - loss: 0.2040 - acc: 0.6848 - val_loss: 0.1939 - val_acc: 0.7607
Epoch 13/20
1377/1377 [==============================] - 36s - loss: 0.1986 - acc: 0.6848 - val_loss: 0.1886 - val_acc: 0.7607
Epoch 14/20
1377/1377 [==============================] - 37s - loss: 0.1936 - acc: 0.6848 - val_loss: 0.1836 - val_acc: 0.7607
Epoch 15/20
1377/1377 [==============================] - 36s - loss: 0.1890 - acc: 0.6848 - val_loss: 0.1791 - val_acc: 0.7607
Epoch 16/20
1377/1377 [==============================] - 37s - loss: 0.1848 - acc: 0.6848 - val_loss: 0.1749 - val_acc: 0.7607
Epoch 17/20
1377/1377 [==============================] - 37s - loss: 0.1808 - acc: 0.6848 - val_loss: 0.1710 - val_acc: 0.7607
Epoch 18/20
1377/1377 [==============================] - 37s - loss: 0.1772 - acc: 0.6848 - val_loss: 0.1673 - val_acc: 0.7607
Epoch 19/20
1377/1377 [==============================] - 37s - loss: 0.1738 - acc: 0.6848 - val_loss: 0.1640 - val_acc: 0.7607
Epoch 20/20
1377/1377 [==============================] - 37s - loss: 0.1707 - acc: 0.6848 - val_loss: 0.1608 - val_acc: 0.7607
```

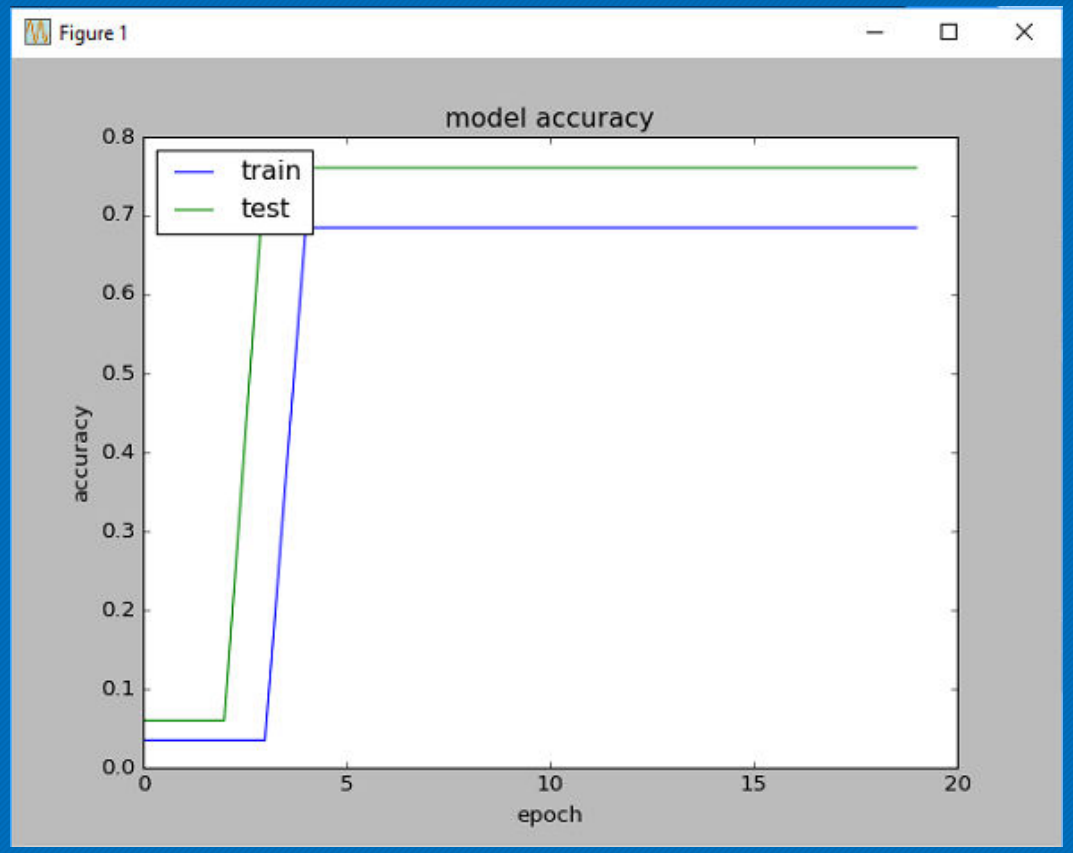
Fig(f) : Output with Accuracy and MSE

Results:

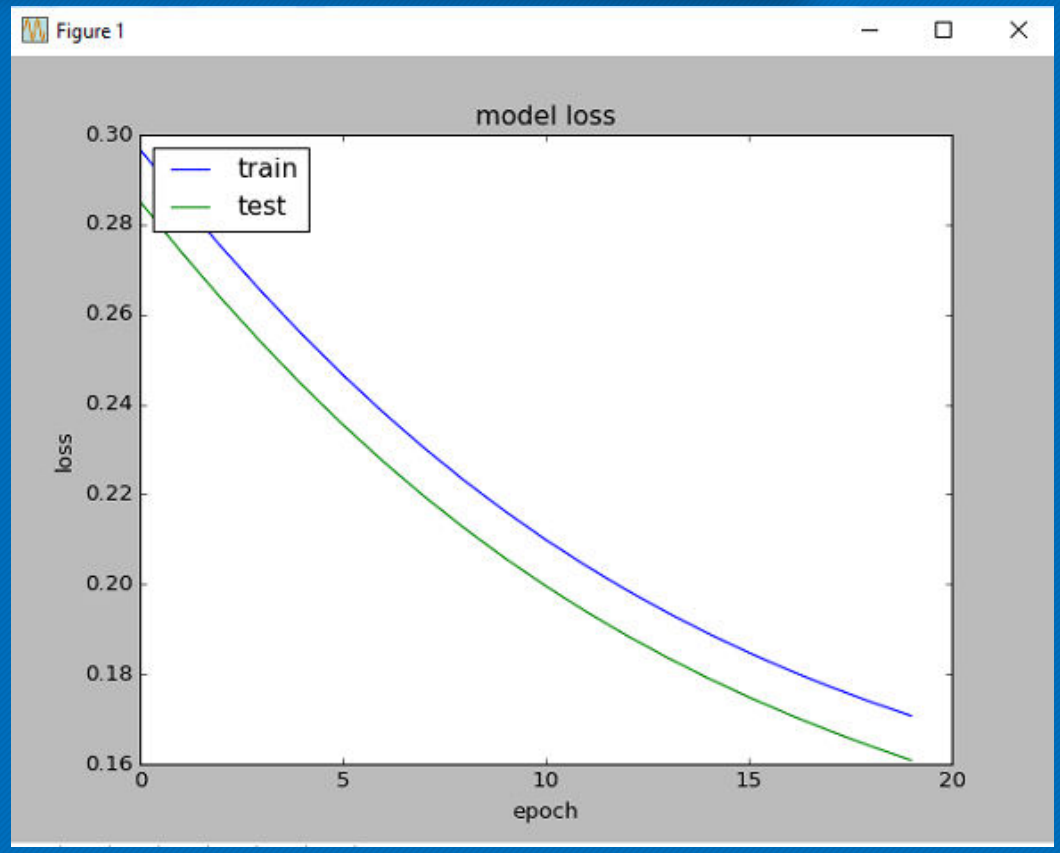
```
('mean squared error :', 0.16084803144137064)
('PREDICTED', array([[ 0.61050826,  0.40708581,  0.34769079,  0.30529341],
       [ 0.61058283,  0.40731785,  0.34763551,  0.30501154],
       [ 0.61047441,  0.40688914,  0.34723935,  0.30520773],
       ...,
       [ 0.61059737,  0.40751094,  0.34674102,  0.3050831 ],
       [ 0.61060041,  0.40754312,  0.34709203,  0.30536228],
       [ 0.61067557,  0.40777513,  0.34703651,  0.30507955]], dtype=float32))
('ORIGINAL', array([[1, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 0, 0, 0],
       ...,
       [1, 0, 0, 0],
       [0, 0, 1, 0],
       [0, 0, 0, 1]]))
```

Fig(g) : Output with Accuracy and MSE

Graphical Representation :



Fig(h) : Accuracy Plot



Fig(i) : Error Plot

Considerations:

- CASE : 4
  - Input Neuron : 21
  - Number of Hidden layers:4
  - Neurons at hidden Layer 1:500
  - Neurons at hidden Layer 2: 400
  - Neurons at hidden Layer 3: 300
  - Neurons at hidden Layer 4 : 200
  - Output Neurons : 4
- Sigmoid function is used as activation function in both Hidden Layer as well as output layer.
- Learning Rate is considered to be : 0.1.
- Momentum :0.7
- Number of Epochs: 20

Results:

```
Epoch 11/20
1377/1377 [==============================] - 58s - loss: 0.2229 - acc: 0.0378 - val_loss: 0.2192 - val_acc: 0.7607
Epoch 12/20
1377/1377 [==============================] - 59s - loss: 0.2195 - acc: 0.6848 - val_loss: 0.2157 - val_acc: 0.7607
Epoch 13/20
1377/1377 [==============================]  - 58s - loss: 0.2162 - acc: 0.6848 - val_loss: 0.2123 - val_acc: 0.7607
Epoch 14/20
1377/1377 [==============================]  - 60s - loss: 0.2131 - acc: 0.6848 - val_loss: 0.2090 - val_acc: 0.7607
Epoch 15/20
1377/1377 [==============================] - 58s - loss: 0.2101 - acc: 0.6848 - val_loss: 0.2059 - val_acc: 0.7607
Epoch 16/20
1377/1377 [==============================] - 58s - loss: 0.2071 - acc: 0.6848 - val_loss: 0.2028 - val_acc: 0.7607
Epoch 17/20
1377/1377 [==============================] - 58s - loss: 0.2043 - acc: 0.6848 - val_loss: 0.1999 - val_acc: 0.7607
Epoch 18/20
1377/1377 [==============================] - 59s - loss: 0.2016 - acc: 0.6848 - val_loss: 0.1971 - val_acc: 0.7607
Epoch 19/20
1377/1377 [==============================] - 58s - loss: 0.1990 - acc: 0.6848 - val_loss: 0.1944 - val_acc: 0.7607
Epoch 20/20
1377/1377 [==============================] - 58s - loss: 0.1965 - acc: 0.6848 - val_loss: 0.1918 - val_acc: 0.7607
```

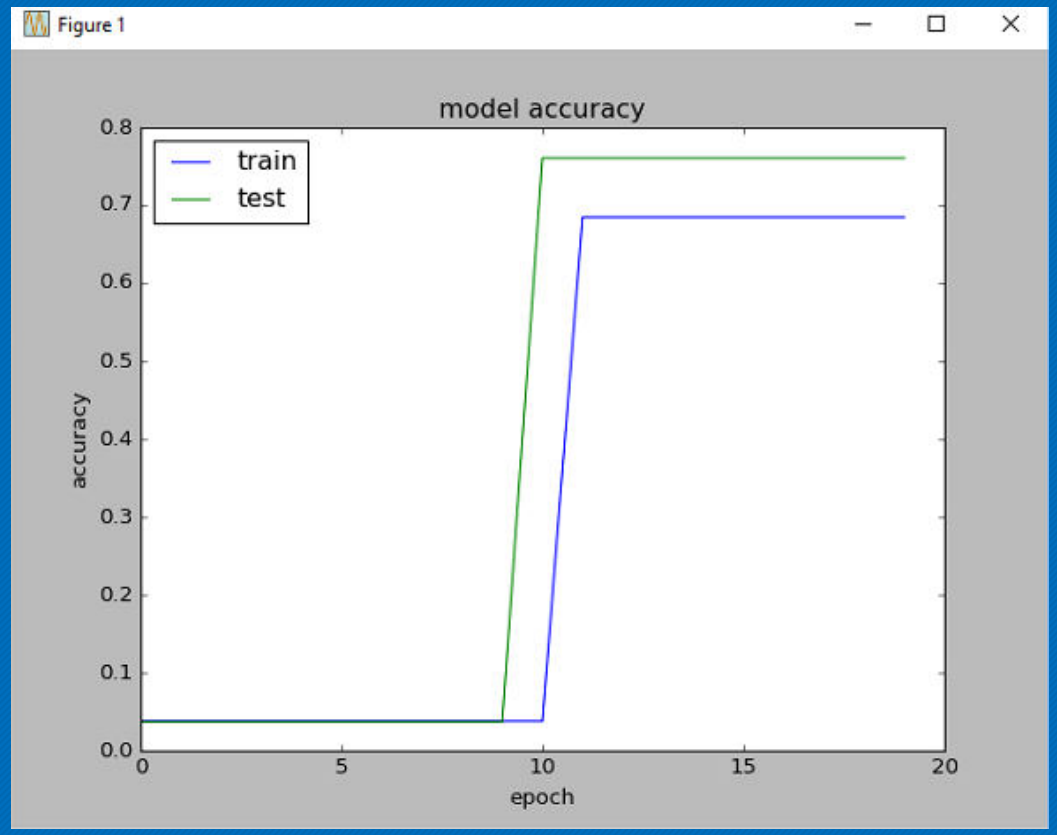
Fig(f) : Output with Accuracy and MSE

Results:

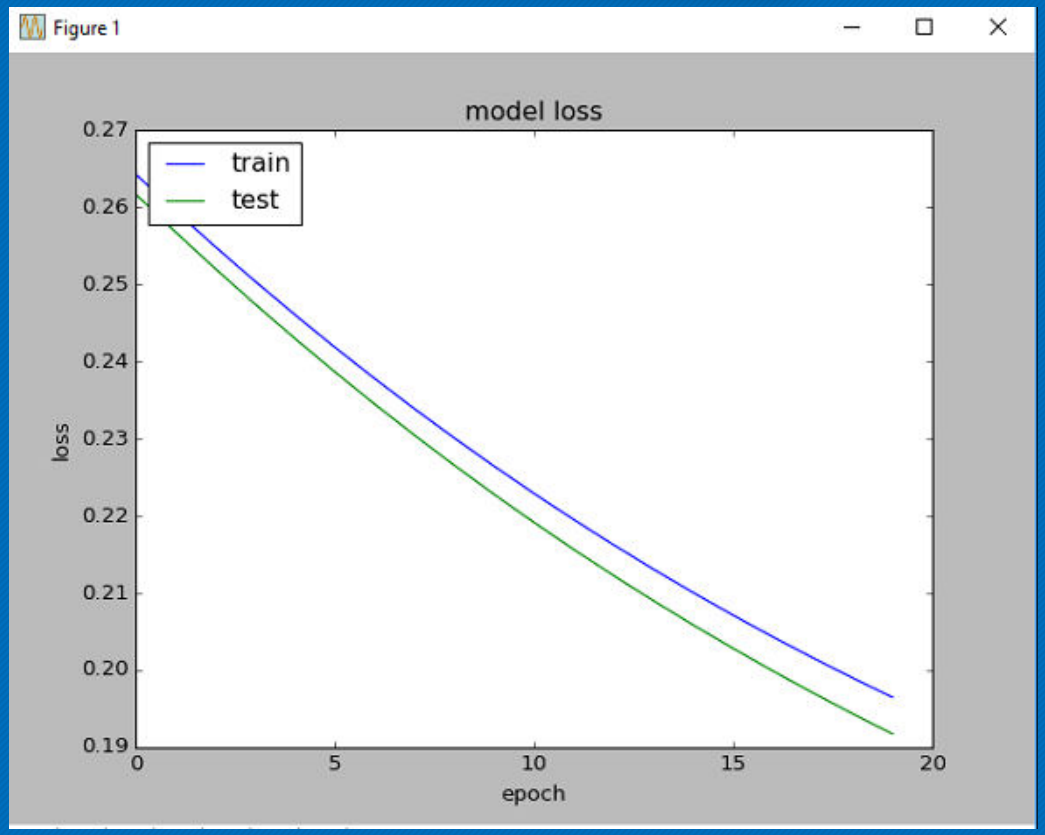
```
('mean squared error :', 0.19175669389572578)
('PREDICTED', array([[ 0.50512058,  0.42121321,  0.35301095,  0.413773  ],
       [ 0.5051195 ,  0.42121616,  0.35300726,  0.41376883],
       [ 0.5051198 ,  0.42120591,  0.35301316,  0.41377863],
       ...,
       [ 0.50511622,  0.42120776,  0.3530184 ,  0.41377339],
       [ 0.50511646,  0.42121091,  0.35301864,  0.41377029],
       [ 0.50511533,  0.42121387,  0.35301504,  0.41376612]], dtype=float32))
('ORIGINAL', array([[1, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 0, 0, 0],
       ...,
       [1, 0, 0, 0],
       [0, 0, 1, 0],
       [0, 0, 0, 1]]))
```

Fig(g) : Output with Accuracy and MSE

Graphical Representation :



Fig(h) : Accuracy Plot



Fig(i) : Error Plot

Future Work:

- We have to add more data items to get more accurate results.
- Need more Optimization Techniques.
- Can use more complex networks.

References:

- Knowledge acquisition and explanation for multi-attribute decision making by M Bohanec, V Rajkovic.
- Machine Learning by Function Decomposition Blaz Zupan, Marko Bohanec, Ivan Bratko, Janez Demsar.
- https://archive.ics.uci.edu/ml/datasets/Car+Evaluation.
- http://neuroph.sourceforge.net/tutorials/carevaluation1/carevaluation1.html