In [1]:
```
from keras.models import Sequential

from keras.layers import Dense
# feed forword

import numpy as np
# to create random dataset

from sklearn.model_selection import train_test_split
```

In [2]:
```
np.random.seed(0)

# to get same list of random numbers every times
```

In [3]:
```
X = np.random.rand(1000, 3)

# vectors will begenerated

# 1000 rows , 3 cols
```

In [4]:
```
X[0:5]

# randomely generated values
```

Out[4]:
```
array([[0.5488135 , 0.71518937, 0.60276338],
       [0.54488318, 0.4236548 , 0.64589411],
       [0.43758721, 0.891773  , 0.96366276],
       [0.38344152, 0.79172504, 0.52889492],
       [0.56804456, 0.92559664, 0.07103606]])
```

In [5]:
```
def true_func(X):
    return np.sin(1.5 * np.pi * X[:, 0]) + np.cos(1.5 * np.pi * X[:,1]) + 2 * X

# X[:, 0] = first col

# sin on first col
# cos on second
# third col * 2
```

In [6]:
```
y = true_func(X)
```

In [7]:
```
y[0:5]

# first 5 values of y
```

Out[7]:
```
array([0.75881293, 1.4217976 , 2.320869  , 1.19872386, 0.24678974])
```

In [8]:
```
from sklearn.model_selection import train_test_split
```

In [9]:
```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random

# random_state=1 = Same random values for training and testing
```

In [10]:
```
model = Sequential()

# supervised
```

In [11]:
```
model.add(Dense(15, input_dim = 3, activation = 'relu'))
```

In [12]:
```
model.add(Dense(1, activation='linear'))

# linear is same as identity
```

In [ ]:
```
# model will give sum of contineous values as output
```

In [18]:
```
model.compile(loss= 'mean_squared_error', optimizer = 'adam', metrics='mean_abs
```

In [19]:
```
model.fit(x_train, y_train , epochs=150, batch_size = 30)
```

```
Epoch 1/150
27/27 [==============================] - 1s 3ms/step - loss: 1.6572 - mean_
absolute_error: 1.0686
Epoch 2/150
27/27 [==============================] - 0s 4ms/step - loss: 1.3319 - mean_
absolute_error: 0.9474
Epoch 3/150
27/27 [==============================] - 0s 3ms/step - loss: 1.1331 - mean_
absolute_error: 0.8648
Epoch 4/150
27/27 [==============================] - 0s 2ms/step - loss: 1.0099 - mean_
absolute_error: 0.8118
Epoch 5/150
27/27 [==============================] - 0s 1ms/step - loss: 0.9225 - mean_
absolute_error: 0.7729
Epoch 6/150
27/27 [==============================] - 0s 1ms/step - loss: 0.8406 - mean_
absolute_error: 0.7366
Epoch 7/150
```

In [20]:
```
predictions = model.predict(x_test)
```

```
7/7 [==============================] - 0s 2ms/step
```

In [22]:
```
from sklearn.metrics import mean_absolute_error
```

In [24]:
```
mean_absolute_error(y_test, predictions)
```

Out[24]:
```
0.18789555132200106
```

In [ ]: