In [1]:
```
### Importing the libraries
import numpy as np
import pandas as pd
```

In [2]:
```
import tensorflow as tf
import keras
```

Part 1 - Data Preprocessing

In [3]:
```
### Importing the dataset
dataset = pd.read_excel('energy.xlsx')
dataset.head()
```

Out[3]:

	AT	V	AP	RH	PE
0	14.96	41.76	1024.07	73.17	463.26
1	25.18	62.96	1020.04	59.08	444.37
2	5.11	39.40	1012.16	92.14	488.56
3	20.86	57.32	1010.24	76.64	446.48
4	10.82	37.50	1009.23	96.62	473.90

In [4]:
```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

In [5]:
```
### Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Part 2 - Building the ANN

In [6]:
```
### Initializing the ANN
ann = tf.keras.models.Sequential()
```

In [7]:
```
### Adding the input layer and the first hidden layer
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

In [8]:
```
### Adding the second hidden layer
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
```

In [9]:
```
### Adding the output layer
ann.add(tf.keras.layers.Dense(units=1))
```

Part 3 - Training the ANN

In [10]:
```
### Compiling the ANN
ann.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

In [11]:
```
### Training the ANN model on the Training set
ann.fit(X_train, y_train, batch_size = 32, epochs = 100)
```

```
Epoch 1/100
240/240 [==============================] - 3s 1ms/step - loss: 3567.5293
Epoch 2/100
240/240 [==============================] - 0s 833us/step - loss: 274.4628
Epoch 3/100
240/240 [==============================] - 0s 876us/step - loss: 251.2420
Epoch 4/100
240/240 [==============================] - 0s 1ms/step - loss: 226.1761
Epoch 5/100
240/240 [==============================] - 0s 679us/step - loss: 203.1667
Epoch 6/100
240/240 [==============================] - 0s 734us/step - loss: 175.8524
Epoch 7/100
240/240 [==============================] - 0s 702us/step - loss: 156.9326
Epoch 8/100
240/240 [==============================] - 0s 720us/step - loss: 136.2898
Epoch 9/100
240/240 [==============================] - 0s 778us/step - loss: 116.6822
Epoch 10/100
```

In [12]:
```
### Predicting the results of the Test set
y_pred = ann.predict(X_test)
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[[430.99 431.23]
 [461.97 460.01]
 [465.47 461.14]
 ...
 [472.66 473.26]
 [439.58 438.  ]
 [458.71 463.28]]
```