In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
import tensorflow as tf
import keras
```

In [3]:

```
tf.__version__
```

Out[3]:

```
'2.9.1'
```

In [4]:

```
keras.__version__
```

Out[4]:

```
'2.9.0'
```

Installing Tensorflow

- pip install Tensorflow --> in anaconda prompt
- pip install tensorflow --> in jupyter notebook

Installing Keras

- pip install Keras --> in anaconda prompt
- !pip install --upgrade keras --> in jupyter notebook

In [5]:

```
df = pd.read_excel('energy.xlsx')
df.head()
```

Out[5]:

	Temperature	Pressure	Humidity	Vaccum	Energy
0	14.96	1024.07	73.17	41.76	463.26
1	25.18	1020.04	59.08	62.96	444.37
2	5.11	1012.16	92.14	39.40	488.56
3	20.86	1010.24	76.64	57.32	446.48
4	10.82	1009.23	96.62	37.50	473.90

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9568 entries, 0 to 9567
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Temperature  9568 non-null   float64
 1   Pressure     9568 non-null   float64
 2   Humidity     9568 non-null   float64
 3   Vaccum       9568 non-null   float64
 4   Energy       9568 non-null   float64
dtypes: float64(5)
memory usage: 373.9 KB
```

In [7]:

```
X = df.drop(columns='Energy')
y = df["Energy"]
```

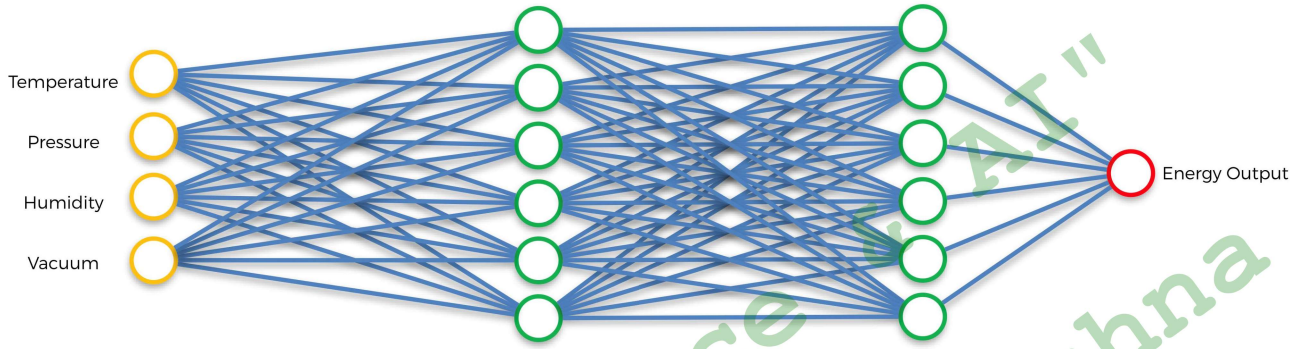
In [8]:

```
### 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)
```

In [9]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

Part 2 - Building the ANN



In [10]:

```
from keras.models import Sequential
from keras.layers import Dense
```

In [11]:

```
def ann_model_regressor():

    ### Initializing the ANN
    model=Sequential()

    ### Adding the input layer and the first hidden layer
    model.add(Dense(input_dim=4, units=6, activation='relu',kernel_initializer="uniform"))

    ### Adding the second hidden layer
    model.add(Dense(units=6, activation='relu',kernel_initializer="uniform"))

    ### Adding the output layer
    model.add(Dense(units=1, activation='relu',kernel_initializer="uniform"))

    ### Compiling the ANN
    model.compile(optimizer = 'adam', loss = 'mean_squared_error')

    return model
```

In [12]:

```
def ann_model_regressor():
    model=Sequential()
    model.add(Dense(input_dim=4, units=6, activation='relu',kernel_initializer="uniform"))
    model.add(Dense(units=6, activation='relu',kernel_initializer="uniform"))
    model.add(Dense(units=1, activation='relu',kernel_initializer="uniform"))
    model.compile(optimizer = 'adam', loss = 'mean_squared_error')
    return model
```

In [13]:

```
ann = ann_model_regressor()
```

In [14]:

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

```
240/240 [==============================] - 0s 1ms/step - loss: 21.4137
Epoch 93/100
240/240 [==============================] - 0s 1ms/step - loss: 21.3468
Epoch 94/100
240/240 [==============================] - 0s 2ms/step - loss: 21.4213
Epoch 95/100
240/240 [==============================] - 0s 1ms/step - loss: 21.4685
Epoch 96/100
240/240 [==============================] - 0s 1ms/step - loss: 21.5556
Epoch 97/100
240/240 [==============================] - 0s 2ms/step - loss: 21.3441
Epoch 98/100

240/240 [==============================] - 0s 1ms/step - loss: 21.3730
Epoch 99/100
240/240 [==============================] - 0s 1ms/step - loss: 21.4444
Epoch 100/100
240/240 [==============================] - 0s 1ms/step - loss: 21.4392
```

Out[14]:

In [15]:

```
### Predicting the results
ypred_train = ann.predict(X_train)
ypred_test = ann.predict(X_test)
```

```
240/240 [==============================] - 0s 1ms/step
60/60 [==============================] - 0s 1ms/step
```

In [16]:

```
from sklearn.metrics import mean_squared_error
print("MSE for train data: ",mean_squared_error(y_train,ypred_train))
print("MSE for test data: ",mean_squared_error(y_test,ypred_test))
```

```
MSE for train data:  21.38453291974682
MSE for test data:  20.07332777085184
```

In [17]:

```
from sklearn.metrics import r2_score
print("R2 for train data: ",r2_score(y_train,ypred_train))
print("R2 for test data: ",r2_score(y_test,ypred_test))
```

```
R2 for train data:  0.9264943515290983
R2 for test data:  0.9313703842199963
```