Step 1: Import the iris dataset using load_iris class from sklearn.datasets and split it into training and testing subsets

In [1]:
```
from sklearn.datasets import load_iris
dataset = load_iris()
x = dataset.data
y = dataset.target
```

In [2]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25)
```

Step 2: Build the MLP Model

--> Import MLPClassifier from sklearn.neural_network module

--> Multilayer Perceptron Classifier

--> Some Hyper Parameters:

```
* max_iter - Maximum number of iterations. Default is 200.

* hidden_layer_sizes - Determines the number of hidden layers and num
ber of neurons in each layer. Default is (100, ) i.e., 1 hidden layer
with 100 neurons.

* activation - Activation function for the hidden layers {'identity',
'logistic', 'tanh', 'relu'}. Default is 'relu'.

* solver - The solver for weight optimization {'lbfgs', 'sgd', 'ada
m'}. Default is 'adam'.

* batch_size - Size of minibatches for stochostic optimizers.

* learning_rate - Learning rate schedule for weight update {'constan
t', 'invscaling', 'adaptive}. Default is 'constant'.

* learning_rate_init - The initial learning rate. Default is 0.001.

* momentum - Momentum for gradient decent update {0 - 1}. Defult is
0.9.
```

--> fit(x_train, y_train) method can be used to build the model.

In [3]:

```
from sklearn.neural_network import MLPClassifier
```

In [4]:

```
# MLP with 100 iterations and one hidden layer with 100 default nodes.

mlp1 = MLPClassifier(max_iter = 100)
mlp1.fit(x_train, y_train)
```

```
C:\Users\admin\.conda\envs\py3_9\lib\site-packages\sklearn\neural_network\_mu
ltilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Maximum
iterations (100) reached and the optimization hasn't converged yet.
  warnings.warn(
```

Out[4]:

```
MLPClassifier(max_iter=100)
```

Step 3: Evaluate the model performance

--> Import accuracy_score function from sklearn.metrics

--> Check for any overfitting

In [5]:

```
from sklearn.metrics import accuracy_score
```

In [6]:

```
print("Training Acuracy:", accuracy_score(y_train, mlp1.predict(x_train)))
print("Test Acuracy:", accuracy_score(y_test, mlp1.predict(x_test)))
```

```
Training Acuracy: 0.9642857142857143
Test Acuracy: 0.8947368421052632
```

Step 4: Fine tune the model performance by channging hyper parammeter values

--> Check for any overfitting

In [7]:

```
# MLP with 100 iterations and two hidden layers with 100 default nodes.

mlp2 = MLPClassifier(max_iter = 100, hidden_layer_sizes = (100, 100))
mlp2.fit(x_train, y_train)

print("Training Acuracy:", accuracy_score(y_train, mlp2.predict(x_train)))
print("Test Acuracy:", accuracy_score(y_test, mlp2.predict(x_test)))
```

```
Training Acuracy: 0.9910714285714286
Test Acuracy: 0.9473684210526315

C:\Users\admin\.conda\envs\py3_9\lib\site-packages\sklearn\neural_network\_mu
ltilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Maximum
iterations (100) reached and the optimization hasn't converged yet.
  warnings.warn(
```