```
Chapter 6 - Other Popular Machine Learning Methods

Part 2 - A neural network with a Perceptron
```

In [1]:
```
import numpy as np
import pandas as pd
import sklearn

from pandas import Series, DataFrame
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
```

In [2]:
```
from sklearn.linear_model import Perceptron
```

In [4]:
```
iris = datasets.load_iris()

x = iris.data
y = iris.target

x[0:10,]
```

Out[4]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1]])
```

In [5]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

In [6]:
```
standardize = StandardScaler()

standardized_x_test = standardize.fit_transform(x_test)

standardized_x_train = standardize.fit_transform(x_train)
```

In [7]:
```
standardized_x_test[0:10,]
```

Out[7]:
```
array([[-0.86168022, -0.77748158,  0.12666374,  0.40214981],
       [-1.40589931,  0.08638684, -1.30609004, -1.38518268],
       [-1.95011839,  0.30235395, -1.49297097, -1.38518268],
       [-0.58957068,  0.73428816, -1.2437964 , -1.38518268],
       [ 0.90703181, -0.56151448,  0.56271924,  0.55109418],
       [ 1.17914135,  0.08638684,  0.43813195,  0.40214981],
       [ 0.63492227, -0.77748158,  0.74960017,  0.99792731],
       [ 0.77097704, -0.34554737,  0.37583831,  0.25320544],
       [ 0.09070318,  0.30235395,  0.68730652,  0.99792731],
       [-0.31746113, -1.2094158 ,  0.12666374, -0.04468331]])
```

In [8]:
```
perceptron = Perceptron(max_iter=50, eta0=0.15, tol=1e-3, random_state=15)

perceptron.fit(standardized_x_train, y_train.ravel())
```

Out[8]:
```
Perceptron(alpha=0.0001, class_weight=None, early_stopping=False, eta0=0.15,
           fit_intercept=True, max_iter=50, n_iter_no_change=5, n_jobs=None,
           penalty=None, random_state=15, shuffle=True, tol=0.001,
           validation_fraction=0.1, verbose=0, warm_start=False)
```

In [9]:
```
y_pred = perceptron.predict(standardized_x_test)
```

In [10]:
```
print(y_pred)
```

```
[1 0 0 0 1 1 2 1 2 1 1 0 2 0 1 0 2 1 1 1 1 0 1 2 0 2 0 2 1 1]
```

In [11]:
```
print(y_test)
```

```
[1 0 0 0 1 1 2 1 1 1 1 0 1 0 2 0 2 2 1 1 1 0 1 2 0 1 0 2 1 2]
```

In [13]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         9
           1       0.79      0.79      0.79        14
           2       0.57      0.57      0.57         7

    accuracy                           0.80        30
   macro avg       0.79      0.79      0.79        30
weighted avg       0.80      0.80      0.80        30
```