# IT3071 – Machine Learning and Optimization Methods

# Lab Sheet 03

## Part A

### 01) Import the dataset

- Which Python library and function will you use to import a dataset suitable for classification?
- How will you separate the features (X) and the target labels (y) from the dataset?
- How can you display:
  - The shape of the dataset
  - The first few rows of feature data
  - The unique target classes

### 02) Split the data into training and testing sets

- Which function will you use to split the dataset into training and testing sets?
- How will you decide the proportion for training and testing data?
- Why might it be important to set a fixed random seed when splitting data?

### 03) Create the MLP Classifier model object

- Which scikit-learn class is used to create an MLP model for classification?
- How can you define the architecture of the hidden layers and the activation function?
- Which parameters control the optimization process and the number of training iterations?

### 04) Train the model with training data

- Which method will you use to train the model?
- What arguments will you pass to this method?

### 05) Check the accuracy of the testing data

- Which method can directly return the accuracy of the model on the test dataset?

- How can you calculate accuracy using the metrics module in scikit-learn?
- What could cause differences between training accuracy and testing accuracy?

## Part B

**01) Import the dataset**

- Which Python library and function will you use to import a dataset suitable for regression?
- How will you separate the features (X) and the target values (y)?
- How can you display:
  - The shape of the dataset
  - The first few rows of feature data
  - The first few target values

**02) Split the data into training and testing sets**

- Which function will you use to split the dataset?
- How will you decide the proportion for training and testing data?

**03) Create the MLP Regressor model object**

- Which scikit-learn class is used to create an MLP model for regression?
- How can you define the architecture of the hidden layers and the activation function?
- Which parameters control the optimization process and the number of training iterations?

**04) Train the model with training data**

- Which method will you use to train the model?
- What arguments will you pass to this method?

**05) Check the error for the testing data**

- Which functions from scikit-learn’s metrics module can be used to measure regression error?

- How would you calculate:
    - Mean Squared Error (MSE)
    - Root Mean Squared Error (RMSE)
    - Mean Absolute Error (MAE)
- What does a smaller error value indicate in regression performance?

Part A - Classification

Import the libraries

In [ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
import numpy as np
import matplotlib.pyplot as plt
```

Import the data

In [ ]:

```
data=pd.read_csv("diabetes.CSV")
data.head()
```

Independent and dependent

In [ ]:

```
x=data.iloc[:,:8]
y=data.iloc[:,8]
```

Training and testing

In [ ]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,
                                               test_size=0.2,random_state=0)
```

Model object

In [ ]:

```
clf = MLPClassifier(alpha=0.01,hidden_layer_sizes=(5,3),
                    random_state=1)
```

Training

In [ ]:

```
clf.fit(x_train,y_train)
```

Predictions

In [ ]:

```
y_pred=clf.predict(x_test)
```

Accuracy of the model

In [ ]:

```
accuracy_score(y_test,y_pred)
```

Part B - Regression

Import the libraries

In [ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPRegressor
```

Import the data

In [ ]:
```
data=pd.read_csv("Boston.CSV")
data.head()
```

Independent and dependent

In [ ]:
```
x=data.iloc[:,:12].values
y=data.iloc[:,12].values
```

Training and testing

In [ ]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,
                                               test_size=0.2,random_state=0)
```

Model object

In [ ]:
```
rg = MLPRegressor(alpha=0.01,hidden_layer_sizes=(3,2),
                  random_state=1,max_iter=300,activation='identity')
```

Training

In [ ]:
```
rg.fit(x_train,y_train)
```

Predictions

In [ ]:
```
y_pred=rg.predict(x_test)
```

Prediction error

In [ ]:
```
np.sqrt(mean_squared_error(y_test,y_pred))
```