Worksheet 6

For this worksheet your will train a neural network for a multi-class classification task. The dataset is from UCI. Please read "Data Set Description" here https://archive.ics.uci.edu/ml/datasets/ecoli (https://archive.ics.uci.edu/ml/datasets/ecoli)

Download the data from https://www.kaggle.com/datasets/kannanaikkal/ecoli-uci-dataset (https://www.kaggle.com/datasets/kannanaikkal/ecoli-uci-dataset)

Please answer the questions below.

In [46]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import LabelEncoder as le
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
```

1. Provide a short description of the data. What are the classes?

In [10]:
```
df = pd.read_csv('./ecoli.csv')
df['SITE'].value_counts()
# the classes are the different site types (cp, im, pp, imU, om, omL, ims
# the data also contains 7 feature variables measuring different things
```

Out[10]:
```
SITE
cp     143
im      77
pp      52
imU     35
om      20
omL      5
imS      2
imL      2
Name: count, dtype: int64
```

2. You will notice that the data are somewhat unbalanced. Show that there are some class with 5, 2 and 2 samples. Remove these from the dataset. Use method `dataframe.loc[ ~ dataframe.SITE.isin([ ....]) ]`

In [11]:
```
unbalanced = ['omL', 'imS', 'imL']
df = df.loc[~df['SITE'].isin(unbalanced)]
df['SITE'].value_counts()
```

Out[11]:
```
SITE
cp     143
im      77
pp      52
imU     35
om      20
Name: count, dtype: int64
```

3. Prepare the dataframe for training the model. Create the features `X` from columns 2 to 8, and the targets `y` from column 9. There should be only five classes after removal of the class `imS` (2), `imL` (2), `omL` (5). Use `LabelEncoder().fit_transform(...)` from `sklearn.preprocessing` to change label names into integers.

In [26]:
```
X = df.iloc[:, 2:8]
y = le().fit_transform(df['SITE'])
print(y)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4]
```

4. Use `MLPClassifier` to train the model. Choose `hidden_layer_sizes`, `activation`, `max_iter`.

In [27]:
```
model = MLPClassifier(hidden_layer_sizes=(20,20), activation='relu', max_
```

5. Since the dataset is too small, we will not do train-test split. We will use all the data and we will assess the model using `confusion_matrix` and `.score` method.

In [37]:
```
model.fit(X, y)
y_pred = model.predict(X)
print(model.score(X, y))
matrix = confusion_matrix(y, y_pred)
print(matrix)
```

```
0.8440366972477065
[[141   0   0   0   2]
 [  2  73   0   0   2]
 [  1  34   0   0   0]
 [  0   0   0  18   2]
 [  6   2   0   0  44]]

/Users/Alex_1/anaconda3/lib/python3.11/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic Opti
mizer: Maximum iterations (500) reached and the optimization hasn't con
verged yet.
  warnings.warn(
```

6. Play with the hyper-parameters to see if you can get better results.

In [44]:
```
model = MLPClassifier(hidden_layer_sizes=(50,50), activation='relu', sol
model.fit(X, y)
y_pred = model.predict(X)
print(model.score(X, y))
matrix = confusion_matrix(y, y_pred)
print(matrix)
```

```
0.9388379204892966
[[143   0   0   0   0]
 [  0  70   7   0   0]
 [  0  11  24   0   0]
 [  0   0   0  20   0]
 [  2   0   0   0  50]]

/Users/Alex_1/anaconda3/lib/python3.11/site-packages/sklearn/neural_net
work/_multilayer_perceptron.py:546: ConvergenceWarning: lbfgs failed to
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html (http
s://scikit-learn.org/stable/modules/preprocessing.html)
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_ite
r)
```

7. Redo using nearest neighbor classifier `KNeighborsClassifier` . You will need to play with k to get the best result.

In [54]:
```
possible_k = [2, 3, 5, 7, 10, 15, 20, 25, 30, 35, 40, 45]
best_score = -1
best_k = -1
for k in possible_k:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X, y)
    score = knn.score(X,y)
    if score > best_score:
        best_score = score
        best_k = k

knn_best = KNeighborsClassifier(n_neighbors=best_k)
knn_best.fit(X, y)
y_pred = knn_best.predict(X)
conf_matrix = confusion_matrix(y, y_pred)
print(conf_matrix)
print(knn_best.score(X,y))
print(best_k)
```

```
[[143   0   0   0   0]
 [  2  75   0   0   0]
 [  2  24   9   0   0]
 [  0   0   1  19   0]
 [  5   2   0   2  43]]
0.8837920489296636
2
```

If KNN score is better than Neural Network classifier score, redo `MLPClassifier` with different number of layers and activation. Try to get close or better than KNN.

8. Give a short reflection on this worksheet.

In [ ]:
```
# This worksheet was interesting to see how playing with hyperparameters
# It was also cool to test neural network performance against knn and vi
```