Session 4

In this session, we will be using one of the classification tasks found in OpenML as the basis for a mock-up exam. More precisely, the task *bank marketing* (data_id=1461) is selected. The classification goal of this task is to predict whether the client will subscribe a term deposit. The input features are numeric (age, account balance, etc.) and nominal (job, marital, education, etc.) types.

Below you can find a baseline result achieved with the logistic regression classifier using default parameters devoting 90% to training and 10% to test (random_state=23).

In [1]:
```
import warnings; warnings.filterwarnings("ignore"); import numpy as np
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

data_id = 1461
test_size = 0.1
X, y = fetch_openml(data_id=data_id, return_X_y=True, as_frame=False, parser="liac-arff")
# Default parameter values: tol=1e-4, C=1e0, solver='lbfgs', max_iter=1e2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=23)
clf = LogisticRegression(random_state=23).fit(X_train, y_train)
print(f'Test error: {(1 - accuracy_score(y_test, clf.predict(X_test)))*100:5.1f}%')
```

```
Test error:  10.8%
```

Exercise 1

Applying the logistic regression classifier with default parameter values except for the parameter C, explore the values of the parameter C in logarithmic scale to determine an optimal value. Report classification error rate on training and test sets. Use random_state=23.

In [2]:
```
print('   solver     tol        C max_iter   etr   ete')
print('--------- ------- ------- -------- ----- -----')
for solver in ['lbfgs']:
    for tol in [1e-4]:
        for C in [1e-3, 1e-2, 1e-1, 1e0, 1e1, 1e2, 1e3]:
            for max_iter in [100]:
                clf = LogisticRegression(solver=solver, tol=tol, C=C, max_iter=max_iter, random_state=23).fit(X_trai
                etr = 1 - accuracy_score(y_train, clf.predict(X_train))
                ete = 1 - accuracy_score(y_test, clf.predict(X_test))
                print(f'{solver:>9} {tol:.1e} {C:.1e} {max_iter:8d} {etr:5.1%} {ete:5.1%}')
```

```
   solver     tol       C max_iter   etr   ete
--------- ------- ------- -------- ----- -----
    lbfgs 1.0e-04 1.0e-03      100 11.3% 11.0%
    lbfgs 1.0e-04 1.0e-02      100 11.3% 10.8%
    lbfgs 1.0e-04 1.0e-01      100 11.3% 10.8%
    lbfgs 1.0e-04 1.0e+00      100 11.3% 10.8%
    lbfgs 1.0e-04 1.0e+01      100 11.3% 10.8%
    lbfgs 1.0e-04 1.0e+02      100 11.3% 10.8%
    lbfgs 1.0e-04 1.0e+03      100 11.3% 10.6%
```

Exercise 2

Applying the logistic regression classifier with default parameter values except for maximum number of iterations and the parameter C that should be fixed to the best value determined in exercise 1, explore the maximum number of iterations in logarithmic scale to determine an optimal value. Report classification error rate on training and test sets. Use random_state=23.

In [3]:

```
print('   solver     tol         C max_iter   etr   ete')
print('--------- ------- ------- -------- ----- -----')
for solver in ['lbfgs']:
    for tol in [1e-4]:
        for C in [1e1]:
            for max_iter in [100, 200, 500, 1000, 2000, 5000, 10000]:
                clf = LogisticRegression(solver=solver, tol=tol, C=C, max_iter=max_iter, random_state=23).fit(X_trai
                etr = 1 - accuracy_score(y_train, clf.predict(X_train))
                ete = 1 - accuracy_score(y_test, clf.predict(X_test))
                print(f'{solver:>9} {tol:.1e} {C:.1e} {max_iter:8d} {etr:5.1%} {ete:5.1%}')
```

```
   solver     tol       C max_iter   etr   ete
--------- ------- ------- -------- ----- -----
    lbfgs 1.0e-04 1.0e+01      100 11.3% 10.8%
    lbfgs 1.0e-04 1.0e+01      200 11.2% 10.5%
    lbfgs 1.0e-04 1.0e+01      500 11.2% 10.4%
    lbfgs 1.0e-04 1.0e+01     1000 11.1% 10.1%
    lbfgs 1.0e-04 1.0e+01     2000 11.1% 10.2%
    lbfgs 1.0e-04 1.0e+01     5000 11.0% 10.2%
    lbfgs 1.0e-04 1.0e+01    10000 11.0% 10.2%
```

Exercise 3

Applying the logistic regression classifier with default parameter values except for the solver, the maximum number of iterations and the parameter C. Use the optimal value for the maximum number of iterations and parameter C determined in the previous experiments. Explore different solvers. Report classification error rate on training and test sets. Use random_state=23.

In [4]:
```
print('         solver     tol       C max_iter   etr   ete')
print('--------------- ------- ------- -------- ----- -----')
for solver in ['lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga']:
    for tol in [1e-4]:
        for C in [1e1]:
            for max_iter in [5000]:
                clf = LogisticRegression(solver=solver, tol=tol, C=C, random_state=23, max_iter=max_iter).fit(X_trai
                etr = 1 - accuracy_score(y_train, clf.predict(X_train))
                ete = 1 - accuracy_score(y_test, clf.predict(X_test))
                print(f'{solver:>15} {tol:.1e} {C:.1e} {max_iter:8d} {etr:5.1%} {ete:5.1%}')
```

```
         solver     tol       C max_iter   etr   ete
--------------- ------- ------- -------- ----- -----
          lbfgs 1.0e-04 1.0e+01     5000 11.0% 10.2%
      liblinear 1.0e-04 1.0e+01     5000 11.0% 10.3%
      newton-cg 1.0e-04 1.0e+01     5000 11.0% 10.2%
newton-cholesky 1.0e-04 1.0e+01     5000 11.0% 10.2%
            sag 1.0e-04 1.0e+01     5000 11.3% 10.8%
           saga 1.0e-04 1.0e+01     5000 11.3% 10.8%
```

Exercise 4

According to the results you have obtained, could you claim that this task is linearly separable? why?

No, the classification error rates achieved are far from zero, so I suspect that this task is not linearly separable.