demo_cross_validation

January 19, 2022

1 Démo : Validation croisée

2 Importer les données

Les données sont disponibles dans le fichier `breast_cancer.csv` dans le répertoire `data`. L'import de données se fait avec la commande `read_csv` de la librairie **pandas**. L'attribut `shape` contient les dimensions de la matrice de données (dataframe).

[1]:
```
import pandas as pd

data = pd.read_csv('../data/breast_cancer.csv', sep=';', index_col='id_sample')
print('data', data.shape)
```

```
data (1016, 51)
```

La méthode `head()` permet d'afficher les 5 premières lignes du dataframe.

[2]:
```
data.head()
```

[2]:
```
                      NAT1     BIRC5      BAG1      BCL2     BLVRA     CCNB1  \
id_sample
TCGA-3C-AAAU-01A  7.100449  3.361004  3.972581  4.145669  4.765233  4.788987
TCGA-3C-AALI-01A  3.453640  4.501040  2.720929  1.493020  5.823480  5.281003
TCGA-3C-AALJ-01A  4.455574  4.164643  3.911511  4.191457  5.987255  5.229446
TCGA-3C-AALK-01A  4.297961  3.920234  3.688335  3.894904  5.211594  4.014641
TCGA-4H-AAAK-01A  1.695378  2.950846  4.110014  3.572843  4.317856  3.772768

                     CCNE1      CDC6     CDC20      CDH3  …    GPR160  \
id_sample                                                 …
TCGA-3C-AAAU-01A  2.164814  2.633598  4.131205  0.133455  …  4.150233
TCGA-3C-AALI-01A  2.535437  2.734157  4.176553  0.110023  …  5.561226
TCGA-3C-AALJ-01A  2.267963  3.379961  4.592752  0.236786  …  2.859309
TCGA-3C-AALK-01A  0.951107  1.472950  3.806552  0.062392  …  3.063807
TCGA-4H-AAAK-01A  1.103958  2.338953  3.473484  0.098773  …  3.289418

                     UBE2T     CXXC5      ANLN     CEP55    ACTR3B      MLPH  \
id_sample
TCGA-3C-AAAU-01A  4.106918  5.528618  3.073409  2.669860  1.928460  5.567999
TCGA-3C-AALI-01A  5.648057  4.711309  3.881110  3.357553  1.168684  7.064176
```

```
TCGA-3C-AALJ-01A  5.213461  6.152875  2.697093  2.599436  1.177678  5.222420
TCGA-3C-AALK-01A  4.166154  5.612184  2.645664  2.448027  1.026535  6.225590
TCGA-4H-AAAK-01A  3.437585  4.299617  2.068516  2.152652  1.513181  5.485277

                      NUF2   TMEM45B          pam50
id_sample
TCGA-3C-AAAU-01A  2.536764  0.213597      luminal-A
TCGA-3C-AALI-01A  3.124620  3.946538  HER2-enriched
TCGA-3C-AALJ-01A  3.053335  0.281303      luminal-B
TCGA-3C-AALK-01A  1.717959  3.289543      luminal-A
TCGA-4H-AAAK-01A  1.537125  2.976903      luminal-A

[5 rows x 51 columns]
```

[3]:
```python
data.groupby(['pam50']).size()
```

[3]:
```
pam50
HER2-enriched     82
basal-like       190
luminal-A        543
luminal-B        201
dtype: int64
```

3 Séparer les données d'expression et les étiquettes

[4]:
```python
# Données d'expression de 50 gènes
X = data.select_dtypes('number')
print('X', X.shape)
```

```
X (1016, 50)
```

[5]:
```python
# Etiquettes correspondantes (sous-types moléculaires)
y = data['pam50']
print('y', y.shape)
```

```
y (1016,)
```

4 Créer une validation croisée stratifiée

4.1 Principe

[6]:
```python
from sklearn.model_selection import StratifiedKFold

random_state = 0
cross_validation = StratifiedKFold(n_splits=3, random_state=random_state,␣
 ↪shuffle=True)
```

```
for train_index, test_index in cross_validation.split(X, y):
    X_train = X.iloc[train_index]
    X_test = X.iloc[test_index]
    print('Train', X_train.shape, 'Test', X_test.shape)
```

```
Train (677, 50) Test (339, 50)
Train (677, 50) Test (339, 50)
Train (678, 50) Test (338, 50)
```

4.2 Calcul détaillé complet

[7]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn import metrics

scaler = StandardScaler()
classifier = SVC(kernel='linear', random_state=random_state,␣
 ↪class_weight='balanced')

accuracy = pd.Series(dtype=float)

iteration = 0
for train_index, test_index in cross_validation.split(X, y):

    iteration += 1 # short version of "iteration = iteration + 1"

    # Train dataset
    X_train = X.iloc[train_index]
    y_train = y.iloc[train_index]

    # Test dataset
    X_test = X.iloc[test_index]
    y_test = y.iloc[test_index]

    # Scaled data
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test) # transform only!

    # Train ML classifier
    classifier.fit(X_train_scaled, y_train)

    # Prediction
    y_pred_test = classifier.predict(X_test_scaled)

    # Accuracy
    accuracy_test = metrics.accuracy_score(y_test, y_pred_test)
    accuracy.loc[iteration] = accuracy_test
    # Display
    print('Iteration', iteration, 'Accuracy =', '{:.8f}'.format(accuracy_test))

print('Mean accuracy', '{:.3f}'.format(accuracy.mean()))
```

```
Iteration 1 Accuracy = 0.93510324
Iteration 2 Accuracy = 0.92920354
Iteration 3 Accuracy = 0.95266272
Mean accuracy 0.939
```

4.3 Pipeline

[8]:
```
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score

pipeline = Pipeline([('scaler', scaler), ('classifier', classifier)])

accuracy = cross_val_score(pipeline, X, y, cv=cross_validation)
print(accuracy)
print('Mean test accuracy', '{:.3f}'.format(accuracy.mean()))
```

```
[0.93510324 0.92920354 0.95266272]
Mean test accuracy 0.939
```

[ ]:
```

```