Supervised learning:
pre-processing, tuning & perform. assessment

Universitat de les Illes Balears

Departament de Ciències Matemàtiques i Informàtica

11752 Aprendizaje Automático
11752 Machine Learning
Máster Universitario
en Sistemas Inteligentes

Alberto ORTIZ RODRÍGUEZ

Data preprocessing and train/test split

- Train/test **splitting** (holdout cross validation):

```
X_train, X_test, y_train, y_test =
         train_test_split(X, y, test_size=.2, random_state=100)
```

- Data **preprocessing**:

```
1) scaler = MinMaxScaler()
   X_train_ = scaler.fit_transform(X_train)
   X_test_ = scaler.transform(X_test)

Or

2) X_ = scaler.fit_transform(X)
   X_train, X_test, y_train, y_test =
         train_test_split(X, y, test_size=.2, random_state=100)

# other data transformations: StandardScaler, …
```

```
DO NOT DO THIS
X_train_ = scaler.fit_transform(X_train)
X_test_ = scaler.fit_transform(X_test)
```

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing

Hyperparameter tuning

- Hyperparameter tuning by **grid search** (only main parameters considered):

```
# model is the classifier, whose parameters have to be tuned
gsclf = GridSearchCV(estimator=model, cv=3, param_grid=pg, scoring=sc, refit=True)
cv: cross-validation strategy → 3 repetitions for every combination in param_grid
param_grid: dictionary or list of dictionaries
scoring: string or tuple of strings referring to implemented metrics
         'accuracy', 'f1', 'precision', 'recall', …
refit: refit the best estimator with the entire dataset
```

[https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression]

```
model = LogisticRegression(solver='saga')
parameters = {
    'penalty': ['l1','l2','elasticnet',None], # regularization penalties
    'C': [0.1, 1.0, 10.0],                    # regularization parameter values
    'fit_intercept': [True, False],           # include beta_0 or not in the model
    'l1_ratio': [0.3, 0.5, 0.8],              # balancing parameter for elasticnet
}
gsclf = GridSearchCV(estimator=model, cv=4, param_grid=parameters, scoring='recall')
gsclf.fit(X_train, y_train)
# useful: gsclf.best_estimator_, gsclf.best_params_, gsclf.best_score_,
# gsclf.predict, gsclf.decision_function, etc.
```

4 x 3 x 2 x 3
= 72
x 4 = 288 trains.

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
https://scikit-learn.org/stable/modules/model_evaluation.html

Hyperparameter tuning

- Hyperparameter tuning by **grid search** (only main parameters considered):

```
# model is the classifier, whose parameters have to be tuned
gsclf = GridSearchCV(estimator=model, cv=3, param_grid=pg, scoring=sc, refit=True)
cv: cross-validation strategy → 3 repetitions for every combination in param_grid
param_grid: dictionary or list of dictionaries
scoring: string or tuple of strings referring to implemented metrics
         'accuracy', 'f1', 'precision', 'recall', …
refit: refit the best estimator with the entire dataset
```

[https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression]

```
model = LogisticRegression(solver='saga')
parameters = [
  {'penalty':None, 'fit_intercept':[True,False]},
  {'penalty':['l1','l2'], 'fit_intercept':[True,False], 'C':[0.1,1.0,10.0]},
  {'penalty':['elasticnet’], 'fit_intercept':[True,False], 'C':[0.1,1.0,10.0],
   'l1_ratio':[0.3,0.5,0.8]}
]
gsclf = GridSearchCV(estimator=model, cv=4, param_grid=parameters, scoring='recall')
gsclf.fit(X_train, y_train)
# useful: gsclf.best_estimator_, gsclf.best_params_, gsclf.best_score_,
# gsclf.predict, gsclf.decision_function, etc.
```

2 + 2 x 2 x 3 + 2 x 3 x 3 = 32 x 4 = 128 trains.

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
https://scikit-learn.org/stable/modules/model_evaluation.html

Performance assessment

- Metrics: (https://scikit-learn.org/stable/modules/model_evaluation.html)

```
y_pred = gsclf_.predict(X_test_)
report = classification_report(y_test,  y_pred); print(report)
test_accuracy = accuracy_score(y_test, y_pred)
test_precision = precision_score(y_test, y_pred)
test_recall = recall_score(y_test, y_pred)
test_f1 = f1_score(y_test, y_pred)
```

- Performance assessment by **cross-validation** (only main parameters considered):

```
clf = gsclf_.best_estimator_
clfp = make_pipeline(StandardScaler(), clf)

scoring = ['accuracy', 'recall']
cv = 5

# stratified K-fold, single metric
scores = cross_val_score(clfp, X, y, scoring=scoring[0], cv=cv)

# stratified K-fold, multiple metrics
scores = cross_validate(clfp, X, y, scoring=scoring, cv=cv)
print('accuracy: ', scores['test_accuracy'])
print('avg acc.: ', np.mean(scores['test_accuracy']))
```

https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation

Performance assessment

- Performance assessment by **cross-validation** (only main parameters considered):

```
# define the cross validation strategy
cv = Kfold(n_splits=5, shuffle=False)
cv = StratifiedKFold(n_splits=5, shuffle=False)
cv = RepeatedKfold(n_splits=5, n_repeats=3) # repeats K-fold n times
cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3)
# there are more alternatives, see
# https://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection

scoring = ['accuracy', 'recall']

# only one metric
scores = cross_val_score(clfp, X, y, scoring=scoring[0], cv=cv)

# calculates and returns multiple metrics
scores = cross_validate(clfp, X, y, cv=cv, scoring=scoring)
print('accuracy: ', scores['test_accuracy'])
print('avg acc.: ', np.mean(scores['test_accuracy']))
```

https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation

Scikit-learn modules

- The same applies to **regression models**

- The previous functions belong to the following **scikit-learn** modules:

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler

from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_validate, cross_val_score
from sklearn.model_selection import Kfold
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import RepeatedKfold
from sklearn.model_selection import RepeatedStratifiedKFold

from sklearn.pipeline import make_pipeline

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
```

Supervised learning: pre-processing, tuning & perform. assessment

Universitat de les Illes Balears

Departament de Ciències Matemàtiques i Informàtica

11752 Aprendizaje Automático
11752 Machine Learning
Máster Universitario
en Sistemas Inteligentes

Alberto ORTIZ RODRÍGUEZ