module 불러오기

In [7]:

```
# data load
from sklearn.datasets import load_digits

# train test split
from sklearn.model_selection import train_test_split

# model
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import LogisticRegression

# report
from sklearn.metrics import classification_report
```

digits 데이터 불러오기

In [ ]:

```
digits = load_digits()
print(digits)
```

In [3]:

```
digits.target_names
```

Out[3]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [5]:

```
data = digits.data
target = digits.target
```

train, test 데이터 분리

In [6]:

```
X_train, X_test, y_train, y_test = train_test_split(data, target, test_size=0.2, random_state=7
X_train
```

Out[6]:

```
array([[ 0.,  1., 12., ..., 14.,  1.,  0.],
       [ 0.,  0.,  9., ..., 16.,  2.,  0.],
       [ 0.,  0.,  3., ...,  0.,  0.,  0.],
       ...,
       [ 0.,  0.,  4., ..., 10.,  1.,  0.],
       [ 0.,  0.,  0., ..., 15.,  5.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.]])
```

각 모델들 학습 시키기

In [8]:

```
# DecisionTree
model_tree = DecisionTreeClassifier()
model_tree.fit(X_train, y_train)
```

Out[8]:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [9]:

```
# RandomForest
model_random_forest = RandomForestClassifier()
model_random_forest.fit(X_train, y_train)
```

Out[9]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

In [10]:

```
# SVM
model_svc = SVC()
model_svc.fit(X_train, y_train)
```

Out[10]:

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [11]:

```
# SGD
model_sgd = SGDClassifier()
model_sgd.fit(X_train, y_train)
```

Out[11]:

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='hinge',
              max_iter=1000, n_iter_no_change=5, n_jobs=None, penalty='l2',
              power_t=0.5, random_state=None, shuffle=True, tol=0.001,
              validation_fraction=0.1, verbose=0, warm_start=False)
```

In [13]:

```
# Logistic Regression
model_LR = LogisticRegression()
model_LR.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: Conver
genceWarning: 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 (https://scikit-lear
n.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[13]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

모델 사용해보고 평가하기

In [15]:

```
# Decision Tree
y_pred = model_tree.predict(X_test)
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       0.97      0.91      0.94        33
           1       0.88      0.86      0.87        49
           2       0.91      0.86      0.89        36
           3       0.83      0.85      0.84        34
           4       0.84      0.86      0.85        42
           5       0.94      0.91      0.92        33
           6       0.89      0.97      0.93        35
           7       0.89      0.92      0.91        37
           8       0.70      0.76      0.73        25
           9       0.76      0.72      0.74        36

    accuracy                           0.86       360
   macro avg       0.86      0.86      0.86       360
weighted avg       0.87      0.86      0.86       360
```

In [16]:

```
# Random Forest
y_pred = model_random_forest.predict(X_test)
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98        30
           1       0.98      1.00      0.99        47
           2       1.00      1.00      1.00        34
           3       1.00      0.95      0.97        37
           4       0.98      0.95      0.97        44
           5       0.97      0.97      0.97        32
           6       0.97      1.00      0.99        37
           7       0.97      0.97      0.97        38
           8       0.96      0.96      0.96        27
           9       0.97      0.97      0.97        34

    accuracy                           0.98       360
   macro avg       0.98      0.98      0.98       360
weighted avg       0.98      0.98      0.98       360
```

In [17]:

```
# SVM
y_pred = model_svc.predict(X_test)
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        31
           1       1.00      1.00      1.00        48
           2       1.00      1.00      1.00        34
           3       1.00      1.00      1.00        35
           4       0.98      1.00      0.99        42
           5       1.00      1.00      1.00        32
           6       1.00      1.00      1.00        38
           7       1.00      1.00      1.00        38
           8       0.96      0.96      0.96        27
           9       1.00      0.97      0.99        35

    accuracy                           0.99       360
   macro avg       0.99      0.99      0.99       360
weighted avg       0.99      0.99      0.99       360
```

In [18]:

```
# SGD
y_pred = model_sgd.predict(X_test)
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        31
           1       0.92      0.92      0.92        48
           2       1.00      1.00      1.00        34
           3       0.91      0.97      0.94        33
           4       0.95      1.00      0.98        41
           5       0.94      0.91      0.92        33
           6       0.97      0.93      0.95        40
           7       1.00      0.95      0.97        40
           8       0.89      0.86      0.87        28
           9       0.94      1.00      0.97        32

    accuracy                           0.95       360
   macro avg       0.95      0.95      0.95       360
weighted avg       0.95      0.95      0.95       360
```

In [19]:

```python
# Logistic Regression
y_pred = model_LR.predict(X_test)
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98        30
           1       0.92      0.92      0.92        48
           2       0.94      1.00      0.97        32
           3       0.97      0.97      0.97        35
           4       0.93      0.95      0.94        42
           5       0.97      0.97      0.97        32
           6       0.97      1.00      0.99        37
           7       0.97      0.95      0.96        39
           8       1.00      0.90      0.95        30
           9       1.00      0.97      0.99        35

    accuracy                           0.96       360
   macro avg       0.96      0.96      0.96       360
weighted avg       0.96      0.96      0.96       360
```

DecisionTree 사용하는게 가장 적절해 보인다. 다른 모델들의 평가 지표들을 보면 100%가 나온게 보인다. overfitting 된 것으로 판단되며, DecisionTree를 사용하는 것이 적절해 보인다.

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```