module 불러오기

In [ ]:

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

# 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
```

wine 데이터 불러오기

In [ ]:

```
wine = load_wine()
print(wine)
```

In [ ]:

```
wine.target_names
```

Out[4]:

```
array(['class_0', 'class_1', 'class_2'], dtype='<U7')
```

In [ ]:

```
data = wine.data
target = wine.target
```

train, test 데이터 분리

In [ ]:

```
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([[1.146e+01, 3.740e+00, 1.820e+00, ..., 7.500e-01, 2.810e+00,
        5.620e+02],
       [1.368e+01, 1.830e+00, 2.360e+00, ..., 1.230e+00, 2.870e+00,
        9.900e+02],
       [1.369e+01, 3.260e+00, 2.540e+00, ..., 9.600e-01, 1.820e+00,
        6.800e+02],
       ...,
       [1.305e+01, 1.650e+00, 2.550e+00, ..., 1.120e+00, 2.510e+00,
        1.105e+03],
       [1.184e+01, 8.900e-01, 2.580e+00, ..., 7.900e-01, 3.080e+00,
        5.200e+02],
       [1.247e+01, 1.520e+00, 2.200e+00, ..., 1.160e+00, 2.630e+00,
        9.370e+02]])
```

각 모델들 학습 시키기

In [ ]:

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

Out[7]:

```
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 [ ]:

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

Out[8]:

```
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 [ ]:

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

Out[9]:

```
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 [ ]:

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

Out[10]:

```
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 [ ]:

```
# 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[11]:

```
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 [ ]:

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

```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        10
           1       1.00      0.82      0.90        17
           2       0.82      1.00      0.90         9

    accuracy                           0.92        36
   macro avg       0.91      0.94      0.92        36
weighted avg       0.93      0.92      0.92        36
```

In [ ]:

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

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00        11

    accuracy                           1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36
```

In [ ]:

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

```
              precision    recall  f1-score   support

           0       0.82      0.90      0.86        10
           1       0.79      0.58      0.67        19
           2       0.27      0.43      0.33         7

    accuracy                           0.64        36
   macro avg       0.63      0.64      0.62        36
weighted avg       0.69      0.64      0.65        36
```

In [ ]:

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

```
              precision    recall  f1-score   support

           0       0.82      0.69      0.75        13
           1       0.00      0.00      0.00         0
           2       0.91      0.43      0.59        23

    accuracy                           0.53        36
   macro avg       0.58      0.38      0.45        36
weighted avg       0.88      0.53      0.65        36
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: Unde
finedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in label
s with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:

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

```
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        12
           1       0.93      0.87      0.90        15
           2       0.82      1.00      0.90         9

    accuracy                           0.92        36
   macro avg       0.92      0.93      0.92        36
weighted avg       0.92      0.92      0.92        36
```

Logistic Regression을 사용한다. 처음애는 Decision tree를 사용하려 했지만 overfitting(100%가 나옴)는 경우가 Logistic Regression보다 1가지 더 많았다. accuracy도 비슷하며, 좀 더 안정적으로 나오는 수치를 택해서 Logistic Regression을 택했다.

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```