module 불러오기

In [ ]:

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

# 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
```

breast cancer 데이터 불러오기

In [ ]:

```
breast_cancer = load_breast_cancer()
print(breast_cancer)
```

In [ ]:

```
breast_cancer.feature_names
```

Out[16]:

```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

In [ ]:

```
breast_cancer.target_names
```

Out[3]:

```
array(['malignant', 'benign'], dtype='<U9')
```

In [ ]:

```
data = breast_cancer.data
target = breast_cancer.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[5]:

```
array([[1.242e+01, 1.504e+01, 7.861e+01, ..., 4.052e-02, 2.901e-01,
        6.783e-02],
       [1.825e+01, 1.998e+01, 1.196e+02, ..., 1.932e-01, 3.063e-01,
        8.368e-02],
       [1.454e+01, 2.754e+01, 9.673e+01, ..., 1.712e-01, 4.218e-01,
        1.341e-01],
       ...,
       [1.727e+01, 2.542e+01, 1.124e+02, ..., 1.739e-01, 2.500e-01,
        7.944e-02],
       [1.185e+01, 1.746e+01, 7.554e+01, ..., 9.140e-02, 3.101e-01,
        7.007e-02],
       [1.403e+01, 2.125e+01, 8.979e+01, ..., 7.963e-02, 2.226e-01,
        7.617e-02]])
```

각 모델들 학습 시키기

In [ ]:

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

Out[6]:

```
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[7]:

```
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[8]:

```
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[9]:

```
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[10]:

```
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       0.92      0.88      0.90        41
           1       0.93      0.96      0.95        73

    accuracy                           0.93       114
   macro avg       0.93      0.92      0.92       114
weighted avg       0.93      0.93      0.93       114
```

In [ ]:

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

```
              precision    recall  f1-score   support

           0       0.87      0.94      0.91        36
           1       0.97      0.94      0.95        78

    accuracy                           0.94       114
   macro avg       0.92      0.94      0.93       114
weighted avg       0.94      0.94      0.94       114
```

In [ ]:

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

```
              precision    recall  f1-score   support

           0       0.77      0.97      0.86        31
           1       0.99      0.89      0.94        83

    accuracy                           0.91       114
   macro avg       0.88      0.93      0.90       114
weighted avg       0.93      0.91      0.92       114
```

In [ ]:

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

```
              precision    recall  f1-score   support

           0       0.92      0.77      0.84        47
           1       0.85      0.96      0.90        67

    accuracy                           0.88       114
   macro avg       0.89      0.86      0.87       114
weighted avg       0.88      0.88      0.87       114
```

In [ ]:

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

```
              precision    recall  f1-score   support

           0       0.87      0.94      0.91        36
           1       0.97      0.94      0.95        78

    accuracy                           0.94       114
   macro avg       0.92      0.94      0.93       114
weighted avg       0.94      0.94      0.94       114
```

Random Forest, logistic Regression을 사용한다. 암은 반드시 발견되어야 하고, 한명의 환자도 놓치면 안되며 recall이 안정적인 모델을 택하여야한다.

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
1
```