유방암 데이터를 이용한 머신러닝

필요한 모듈 import

```
# 교차 검증
from sklearn.model_selection import cross_val_score
# 유방암
from sklearn.datasets import load_breast_cancer
# 선형회귀
from sklearn.linear_model import LinearRegression
# KNN
from sklearn.neighbors import KNeighborsClassifier
# SVM
from sklearn.svm import LinearSVC
# 의사결정트리
from sklearn.tree import DecisionTreeClassifier
# 랜덤포레스트
from sklearn.ensemble import RandomForestClassifier

cancer = load_breast_cancer()
cancer.keys() # dict_keys(['data', 'target', 'target_names', 'DESCR',
                           'feature_names', 'filename'])
cancer.target_names # array(['malignant', 'benign'], dtype='<U9')
cancer.target # 0 1

type(cancer) # sklearn.utils.Bunch
```

학습

선형회귀 학습모델

```
lr = LinearRegression()
lr

# LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=
```

KNN학습모델

```
knn = KNeighborsClassifier(n_neighbors=4) # 개수를 정하려면 몇개로 되어있는지 값
                                          # 검증해야한다.
knn

#KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
#                     metric_params=None, n_jobs=None, n_neighbors=4, p=2,
#                     weights='uniform')
```

SVM 학습모델

```
svm = LinearSVC(random_state=0)
svm

#LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
#          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
#          multi_class='ovr', penalty='l2', random_state=0, tol=0.0001,
#          verbose=0)
```

의사결정나무 학습모델

```
tree = DecisionTreeClassifier(max_depth = 3, random_state=0)
tree

#DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
#                       max_depth=3, 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=0, splitter='best')
```

랜덤포레스트 학습모델

```
forest = RandomForestClassifier(n_estimators=6)
forest

#RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
#                       criterion='gini', max_depth=None, max_features='aut
```

```
#                        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=6,
#                        n_jobs=None, oob_score=False, random_state=None,
#                        verbose=0, warm_start=False)
```

교차검증

```
# 선형회귀 학습 후, 교차검증
score1 = cross_val_score(lr, cancer.data, cancer.target)

# KNN 학습 후, 교차검증
score2 = cross_val_score(knn, cancer.data, cancer.target)

# SVM 학습 후, 교차검증
score3 = cross_val_score(svm, cancer.data, cancer.target)

# 의사결정나무 학습 후, 교차검증
score4 = cross_val_score(tree, cancer.data, cancer.target)

# 랜덤포레스트 학습 후, 교차검증
score5 = cross_val_score(forest, cancer.data, cancer.target)

print('선형회귀 교차검증 점수 : {:.2f}'.format(score1.mean())) # 70%
print('KNN 교차검증 점수 : {:.2f}'.format(score2.mean())) # 92%
print('SVM 교차검증 점수 : {:.2f}'.format(score3.mean())) # 82%
print('의사결정트리 교차검증 점수 : {:.2f}'.format(score4.mean())) # 92%
print('랜덤포레스트 교차검증 점수 : {:.2f}'.format(score5.mean())) # 95%
```

학습모델을 이용해서 다시 학습

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(cancer.data,
                                                    cancer.target,
                                                    random_state = 0)

lr = LinearRegression().fit(X_train, Y_train)
```

```
knn = KNeighborsClassifier(n_neighbors=4).fit(X_train, Y_train)

svm = LinearSVC(random_state=0).fit(X_train, Y_train)

tree = DecisionTreeClassifier(max_depth = 3, random_state=0).fit(X_train, Y

forest = RandomForestClassifier(n_estimators=6).fit(X_train, Y_train)

print('선형회귀 정확도 : {:.2f}'.format(lr.score(X_test, Y_test))) # 73%
print('KNN 정확도 : {:.2f}'.format(knn.score(X_test, Y_test))) # 92%
print('SVM 정확도 : {:.2f}'.format(svm.score(X_test, Y_test))) # 87%
print('의사결정트리 정확도 : {:.2f}'.format(tree.score(X_test, Y_test))) # 94%
print('랜덤포레스트 정확도 : {:.2f}'.format(forest.score(X_test, Y_test))) # 9
```

> 성능이 좋아진 것도 있고 나빠진 것도 있다.

> train_test_split 은 데이터를 무작위로 나누는데
> 이럴 경우 무작위로 나뉘어진 셋에 어떤 데이터들이 담기느냐에 따라서 정확도가 높고 낮아질 수 있다.

> 그러나, 교차검증은
> 각 폴드가 한 번씩 테스트 세트가 되므로
> train_test_split 보다 데이터가 편향될 확률은 낮다

> -> 나중에 새로운 데이터가 들어오면 얘가 어떤품종(0인지 1인지 2인지)인지 구별해내려고 학습