

# 사례1: 유방암 세포 진단



# SVM, kNN을 사용해 유방암 진단하기

## ■ 목표

- ✓ UCI 데이터 저장소에서 받은 유방암 데이터를 탐색하고, 머신러닝 알고리즘을 이용하여 세포를 양성 또는 악성으로 예측해 보자

## ■ 순서

- ✓ 준비하기
- ✓ 데이터 전처리와 데이터 탐색
- ✓ 데이터셋 분리 및 학습
- ✓ 머신러닝을 사용한 예측



# SVM, kNN 모델을 사용한 유방암 진단

## ■ 준비하기

- ✓ 필요한 파이썬 라이브러리 임포트
- ✓ (참고) 주요 라이브러리의 버전 확인 방법

```
import scipy
import numpy
import matplotlib
import pandas
import sklearn

pkgs = [scipy, numpy, matplotlib, sklearn]
for pkg in pkgs:
    print(f'{pkg.__name__}: {pkg.__version__}')

scipy: 1.7.3
numpy: 1.21.6
matplotlib: 3.2.2
sklearn: 1.0.2
```

```
import numpy as np
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
import pandas as pd
```

## ■ 데이터셋 로드하기


- ✓ UCI 머신러닝 데이터 저장소 URL을 사용해 바로 로드

```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data'
names = ['id', 'clump_thickness', 'uniform_cell_size', 'uniform_cell_shape', 'marginal_adhesion',
         'single_epithelial_size', 'bare_nuclei', 'bland_chromatin', 'normal_nucleoli', 'mitoses', 'class']
df = pd.read_csv(url, names=names)
```



# Breast Cancer Wisconsin (Diagnostic) Data Set

- Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass
- They describe characteristics of the cell nuclei present in the image
- <https://goo.gl/U2Uwz2>

 **Breast Cancer Wisconsin (Diagnostic)**  
Donated on 10/31/1995

Diagnostic Wisconsin Breast Cancer Database.

<b>Dataset Characteristics</b> Multivariate	<b>Subject Area</b> Life Science	<b>Associated Tasks</b> Classification
<b>Feature Type</b> Real	<b># Instances</b> 569	<b># Features</b> 30

**Dataset Information** ^

**Additional Information**

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. A few of the images can be found at <http://www.cs.wisc.edu/~street/images/...>

SHOW MORE ▾

**Has Missing Values?**  
No



# 데이터 전처리와 데이터 탐색

## ■ 결측값 처리

- ✓ `df.replace()` 함수를 사용해 '?' 처리된 결측값을 -99999로 바꾸기
- ✓ `df.drop()` 함수를 사용해 필요없는 데이터 제거
- ✓ `df.shape`을 이용하여 전체 데이터셋의 구조 확인

## ■ 데이터 탐색

- ✓ `df.head()`를 이용하여 첫 5행에 대한 데이터 확인
- ✓ `df.describe()` 함수를 사용해 각 열에 대한 요약통계량 확인

```
# data preprocessing
df.replace('?', -99999, inplace=True)
df.head()
```

	id	clump_thickness	uniform_cell_size	uniform_cell_shape	marginal_adhesion	single_epithelial_size	bare_nuclei	bland_chromatin
0	1000025	5	1	1	1	2	1	1
1	1002945	5	4	4	5	7	10	10
2	1015425	3	1	1	1	2	2	2
3	1016277	6	8	8	1	3	4	4
4	1017023	4	1	1	3	2	1	1



```
df.drop(columns='id', axis=1, inplace=True)
df.head()
```

	clump_thickness	uniform_cell_size	uniform_cell_shape	marginal_adhesion	single_epithelial_size	bare_nuclei	bland_chromatin
0	5	1	1	1	2	1	1
1	5	4	4	5	7	10	10
2	3	1	1	1	2	2	2
3	6	8	8	1	3	4	4
4	4	1	1	3	2	1	1



```
print(df.shape)
```

(699, 10)

```
df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
clump_thickness	699.0	4.417740	2.815741	1.0	2.0	4.0	6.0	10.0
uniform_cell_size	699.0	3.134478	3.051459	1.0	1.0	1.0	5.0	10.0
uniform_cell_shape	699.0	3.207439	2.971913	1.0	1.0	1.0	5.0	10.0
marginal_adhesion	699.0	2.806867	2.855379	1.0	1.0	1.0	4.0	10.0
single_epithelial_size	699.0	3.216023	2.214300	1.0	2.0	2.0	4.0	10.0
bland_chromatin	699.0	3.437768	2.438364	1.0	2.0	3.0	5.0	10.0
normal_nucleoli	699.0	2.866953	3.053634	1.0	1.0	1.0	4.0	10.0
mitoses	699.0	1.589413	1.715078	1.0	1.0	1.0	1.0	10.0
class	699.0	2.689557	0.951273	2.0	2.0	2.0	4.0	4.0

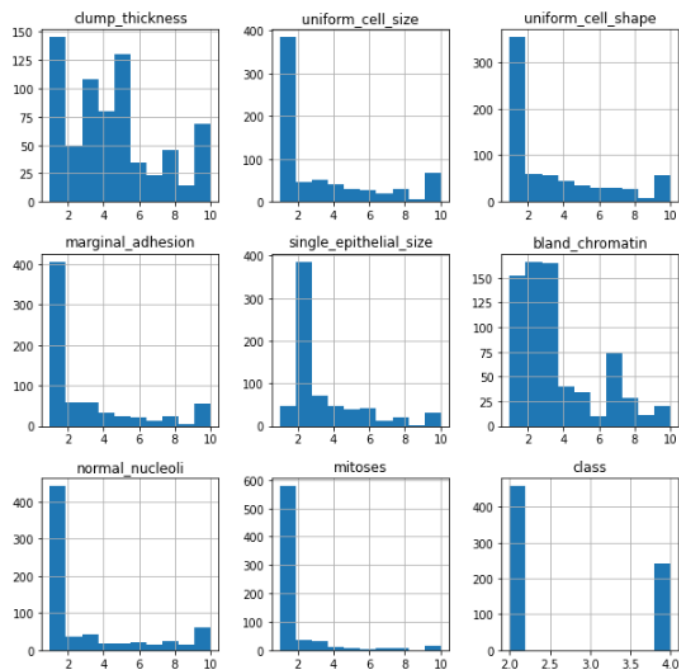


# 데이터 전처리와 데이터 탐색

## ■ 데이터 탐색

- ✓ `df.hist()` 함수를 사용해 각 변수에 대한 히스토그램 출력 후 분포 확인

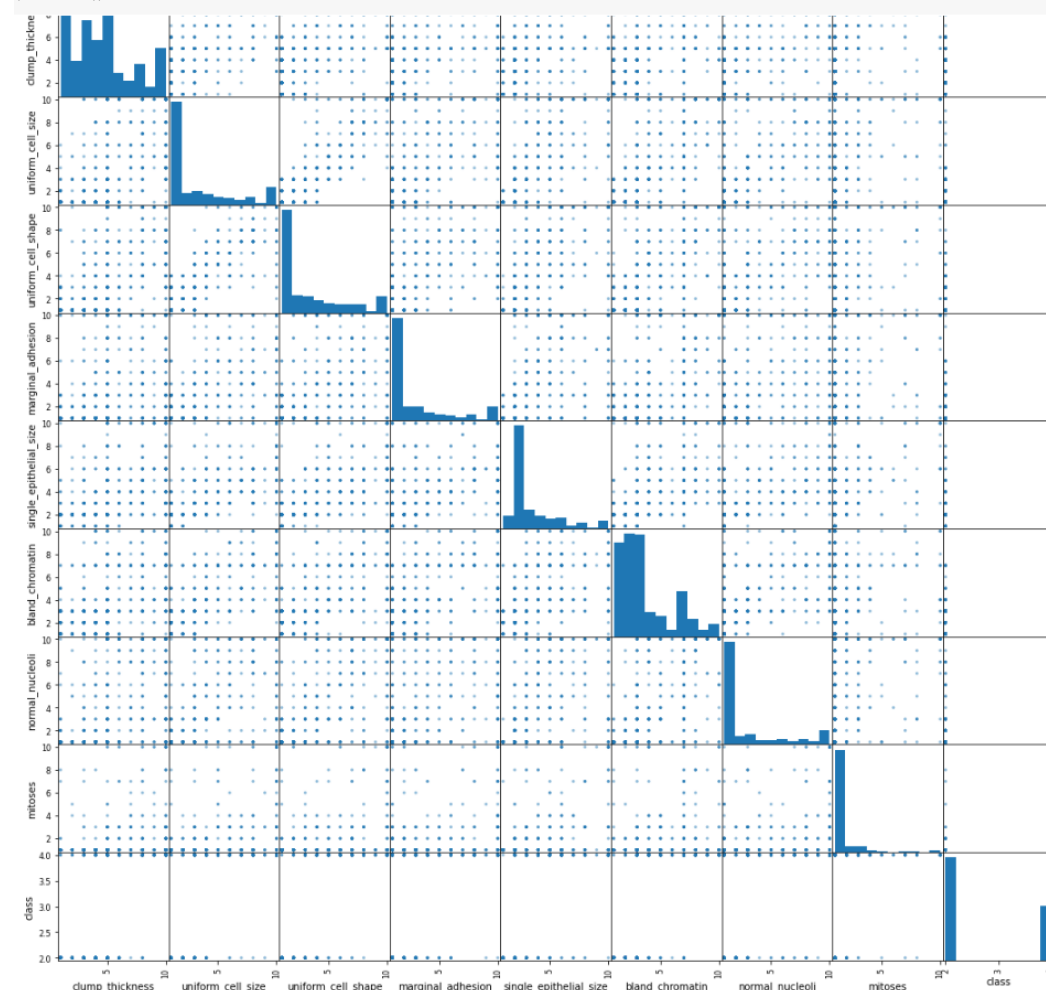
```
df.hist(figsize=(10,10))  
plt.show()
```



## ■ 변수들 간의 관계

- ✓ `scatter_matrix()` 함수를 사용해 산점도 행렬을 생성하고 시각화하여 변수간 관계 확인

```
scatter_matrix(df, figsize=(18,18))  
plt.show()
```



# 데이터셋 분리

- 데이터셋을 훈련용과 테스트용으로 분리
  - ✓ X에는 class 열을 제외한 모든 변수를 포함
  - ✓ y에는 타깃 데이터인 class 열
- train\_test\_split() 함수 사용
  - ✓ test\_size 인자를 통해 전체 데이터셋에서 훈련용으로 사용할 데이터의 양을 설정
  - ✓ 0.2로 지정한다는 것은 전체 데이터의 20%가 훈련용으로 사용
- scikit-learn에서 제공하는 kNN과 SVM 모델에 대한 클래스 사용
  - ✓ 클래스 인스턴스를 models라는 파이썬 리스트에 저장
  - ✓ results, names라는 파이썬 리스트는 나중에 결과값을 보관할 저장소
  - ✓ for문을 사용해 두 모델을 동시에 훈련시키고 k-fold 교차 검증 시행 후 그 결과인 평균값과 표준편차가 출력되게 함
- k-fold 교차 검증
  - ✓ 훈련 데이터를 k에 해당하는 n\_splits개의 서브 그룹으로 나누고, 서브 그룹에서 하나를 제외한 후 학습시킨 다음, 제외했던 나머지 그룹으로 모델을 평가하는 방법
  - ✓ 그 다음에는 제외했던 것을 포함시키고 다른 하나를 제외한 후 모델을 훈련시키고 평가하는 방식을 반복한다

```
X = np.array(df.drop(columns='class',axis=1))
y = np.array(df['class'])

X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.2)

scoring='accuracy'

models = []
models.append(('KNN',KNeighborsClassifier(n_neighbors=5)))
models.append(('SVM',SVC(gamma='auto')))

results=[]
names = []

for name, model in models:
    kfold = model_selection.KFold(n_splits=10)
    cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

KNN: 0.967857 (0.015567)
SVM: 0.957110 (0.016317)
```



# 머신러닝을 사용한 예측

- 학습에 사용하지 않은 데이터에 대한 예측
  - ✓ `model.fit()` 함수를 사용해 훈련 데이터를 가지고 학습 진행
  - ✓ 모델이 학습되면 테스트 데이터인 `X_test`에 대해 `model.predict()` 함수로 예측 진행
- 결과 출력
  - ✓ 모델 이름을 출력하고, 타깃 정보가 들어있는 `y_test`와 모델이 예측한 `predictions`를 서로 비교해 `accuracy_score()`를 실행하여 결과 출력
  - ✓ `classification_report()` 함수를 사용하여 분류 보고서를 출력
    - 거짓 양성, 거짓 음성 등을 확인

```
# prediction
for name, model in models:
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    print(f'{name}: {accuracy_score(y_test, predictions)}')
    print(classification_report(y_test, predictions))
```

KNN: 0.9857142857142858

	precision	recall	f1-score	support
2	0.98	1.00	0.99	96
4	1.00	0.95	0.98	44
accuracy			0.99	140
macro avg	0.99	0.98	0.98	140
weighted avg	0.99	0.99	0.99	140

SVM: 0.9857142857142858

	precision	recall	f1-score	support
2	1.00	0.98	0.99	96
4	0.96	1.00	0.98	44
accuracy			0.99	140
macro avg	0.98	0.99	0.98	140
weighted avg	0.99	0.99	0.99	140





# 하나의 사례에 대한 예측

- 직접 하나의 세포의 특성들을 설정한 다음, 그 세포에 대해 양성, 악성 여부를 판단해 보자
  - ✓ np.array()로 하나의 세포를 표현하는 값을 만든다
    - 사실 원래 데이터의 열 번째 관측 세포의 값임
  - ✓ 모델의 정확도는 98.57%로, 제시한 세포는 양성으로 판정하고 있음
    - 실제 데이터셋에서 이것이 맞다는 것을 확인할 수 있음
- Inclass Assignment
  - ✓ example1 = [[5,3,3,3,2,3,4,4,1]]
  - ✓ example2 = [[7,3,2,10,5,10,5,4,4]]
  - ✓ 위의 두 사례에 대한 예측값을 구해보고 오른쪽 그림과 같은 예측 결과를 캡처하여 업로드한다

```
clf = SVC(gamma='auto')

clf.fit(X_train, y_train)
accuracy = clf.score(X_test, y_test)
print(accuracy)

example = np.array([[4,2,1,1,2,1,2,1,1]])
example = example.reshape(len(example), -1)
prediction = clf.predict(example)
print(prediction)
```

```
0.9857142857142858
[2]
```

```
print(df.loc[9])
```

clump_thickness	4
uniform_cell_size	2
uniform_cell_shape	1
marginal_adhesion	1
single_epithelial_size	2
bare_nuclei	1
bland_chromatin	2
normal_nucleoli	1
mitoses	1
class	2

Name: 9, dtype: object



# 요약

- UCI 데이터 저장소에서 데이터를 임포트해 판다스 데이터프레임으로 불러들인 다음, 데이터 전처리를 거쳐 데이터 탐색을 수행하였음
  - ✓ 그 수행 과정에서 히스토그램이나 산점도 행렬 등을 만들어보았음
- 데이터셋을 훈련 데이터와 테스트 데이터로 나누고, kNN, SVM 분류 모델을 구성하고, 분류 보고서를 통해 두 분류기의 성능을 서로 비교할 수도 있었음
- 마지막으로 임의의 데이터가 주어졌을 때 모델이 세포를 어떻게 분류하는지 예측할 수 있었음

