Python

Matplotlib/빈도분석/kNN

김선녕(sykim.lecture@gmail.com)

참고문헌 : 파이썬을 이용한 빅데이터 수집, 분석과 시각화 - 비팬북스, 이원화

Matplotlib

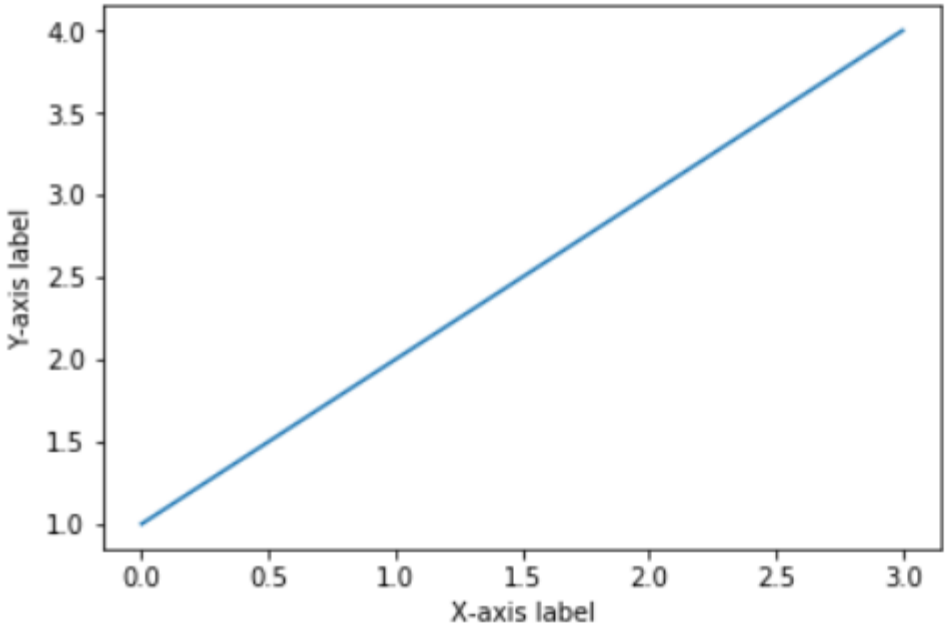
빈도분석

k-최근접 이웃 알고리즘(k-Nearest Neighbors)

기본 그래프 그리기

In [2]:

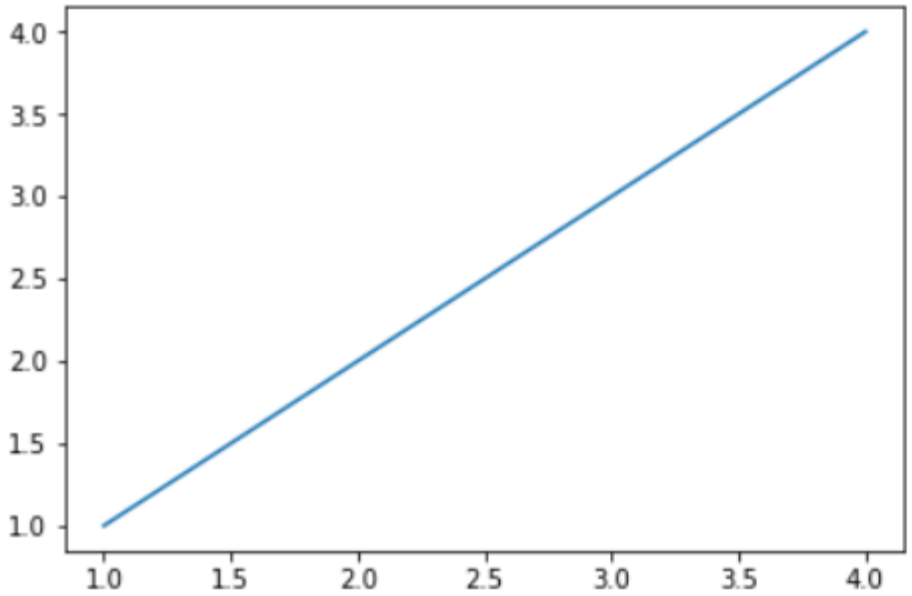
```
1 import matplotlib.pyplot as plt
2 plt.plot([1,2,3,4])
3 plt.xlabel("X-axis label")
4 plt.ylabel("Y-axis label")
5 plt.show()
```



기본 그래프 그리기

In [3]:

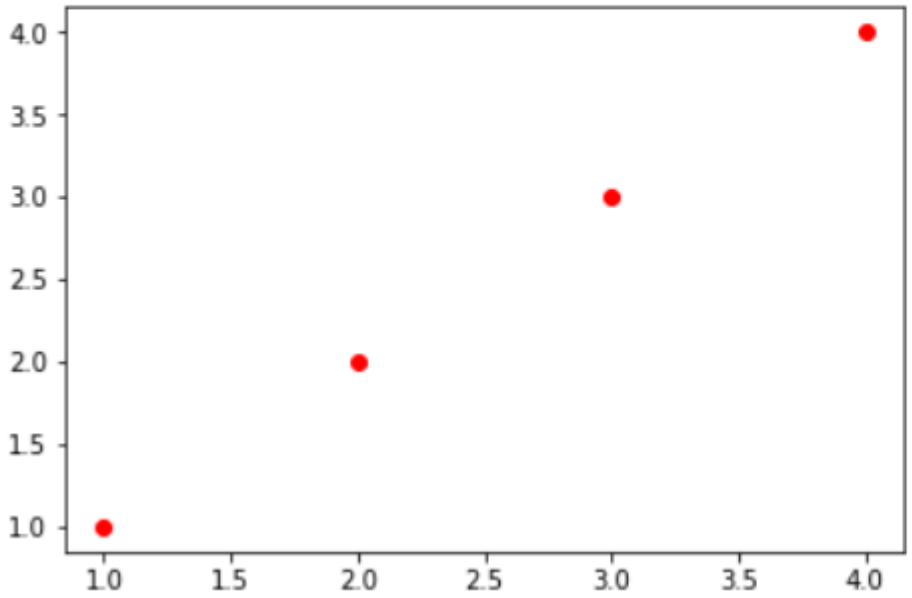
```
1 plt.plot([1,2,3,4],[1,2,3,4])
2 plt.show()
```



기본 그래프 그리기

In [6]:

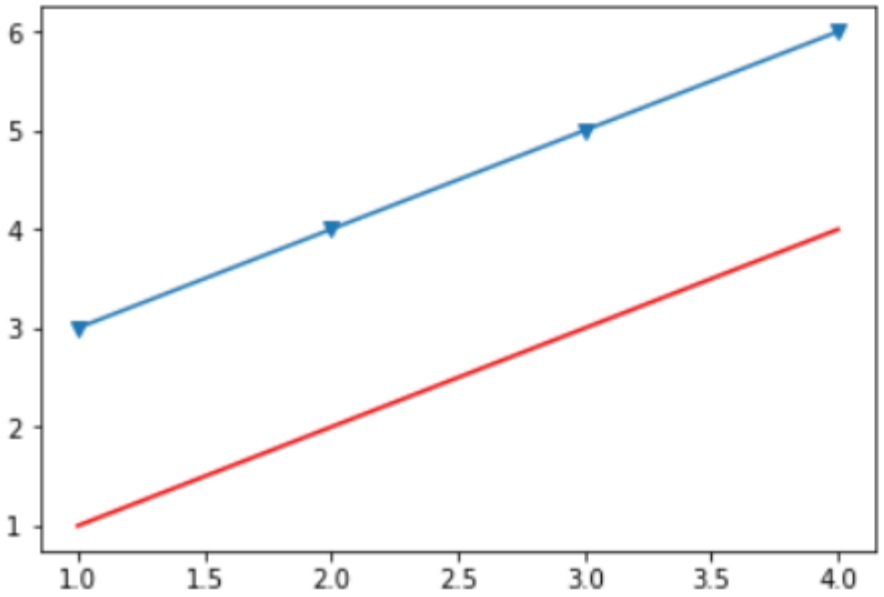
```
1 # 기본값 : 파란색(b) 라인(-)
2 # ro : 적색 o, bv : 파란색 v 마크, 그외 matplotlib 공식사이트에서 확인
3 plt.plot([1,2,3,4],[1,2,3,4],'ro')
4 plt.show()
```



다수의 그래프 그리기

In [7]:
```
1 # 다수의 그래프 그리기
2 plt.plot([1,2,3,4],[1,2,3,4],'r-',[1,2,3,4],[3,4,5,6], 'v-' )
```

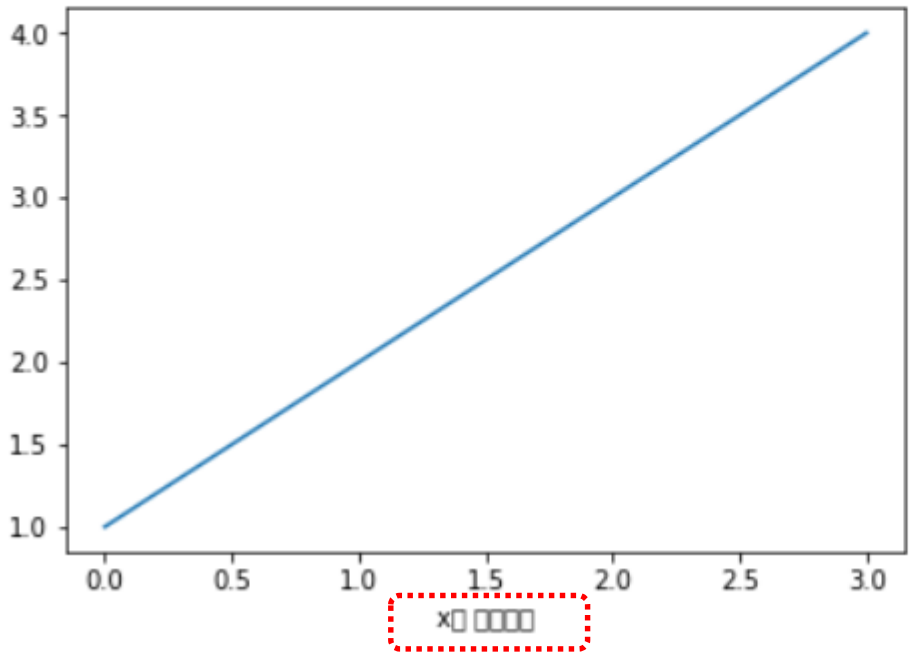
Out[7]:
```
[<matplotlib.lines.Line2D at 0x22df8386828>,
 <matplotlib.lines.Line2D at 0x22df83869e8>]
```



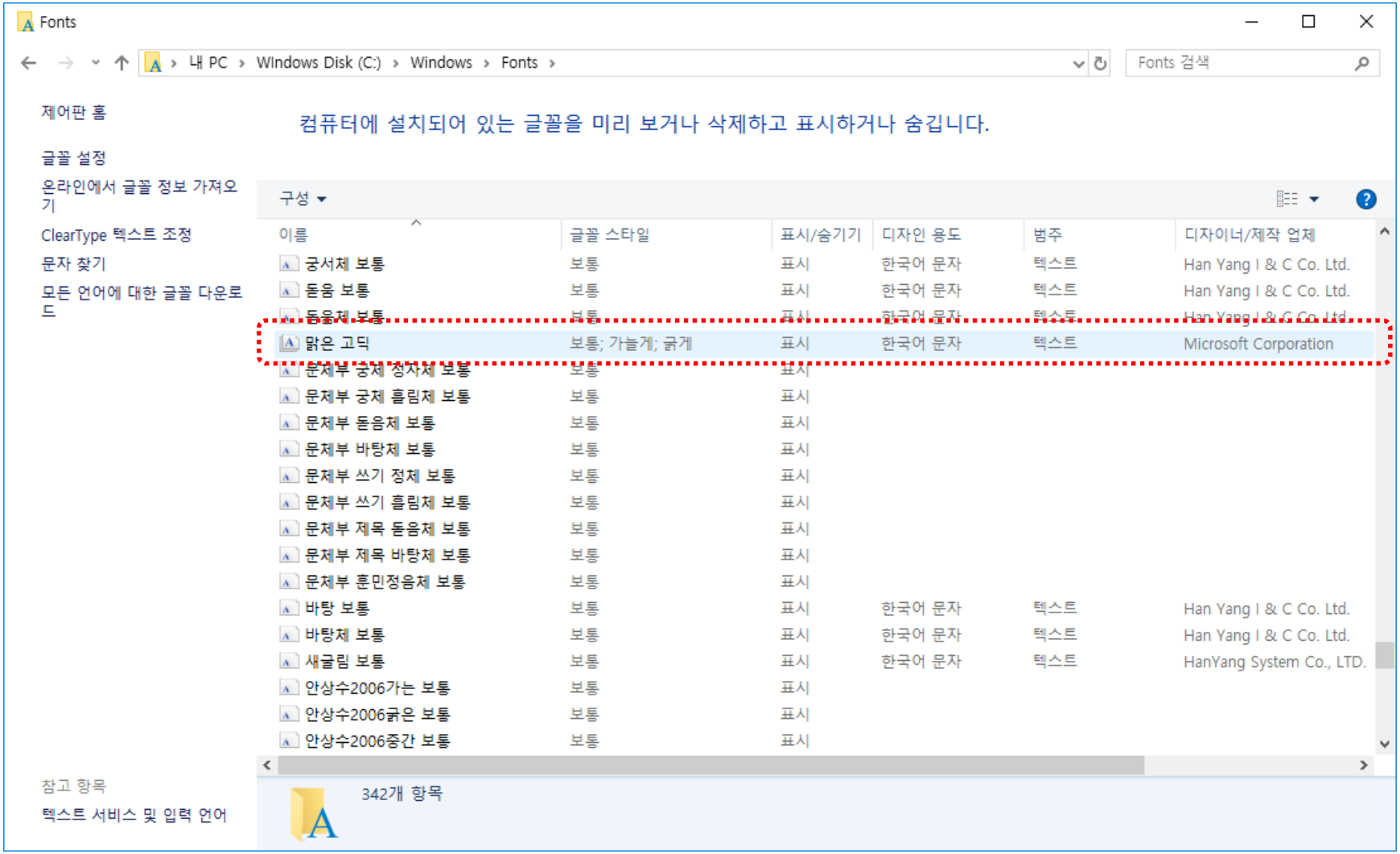
한글처리

In [8]:
```
1 plt.plot([1,2,3,4])
2 plt.xlabel('x축 한글표시')
3 plt.show
```

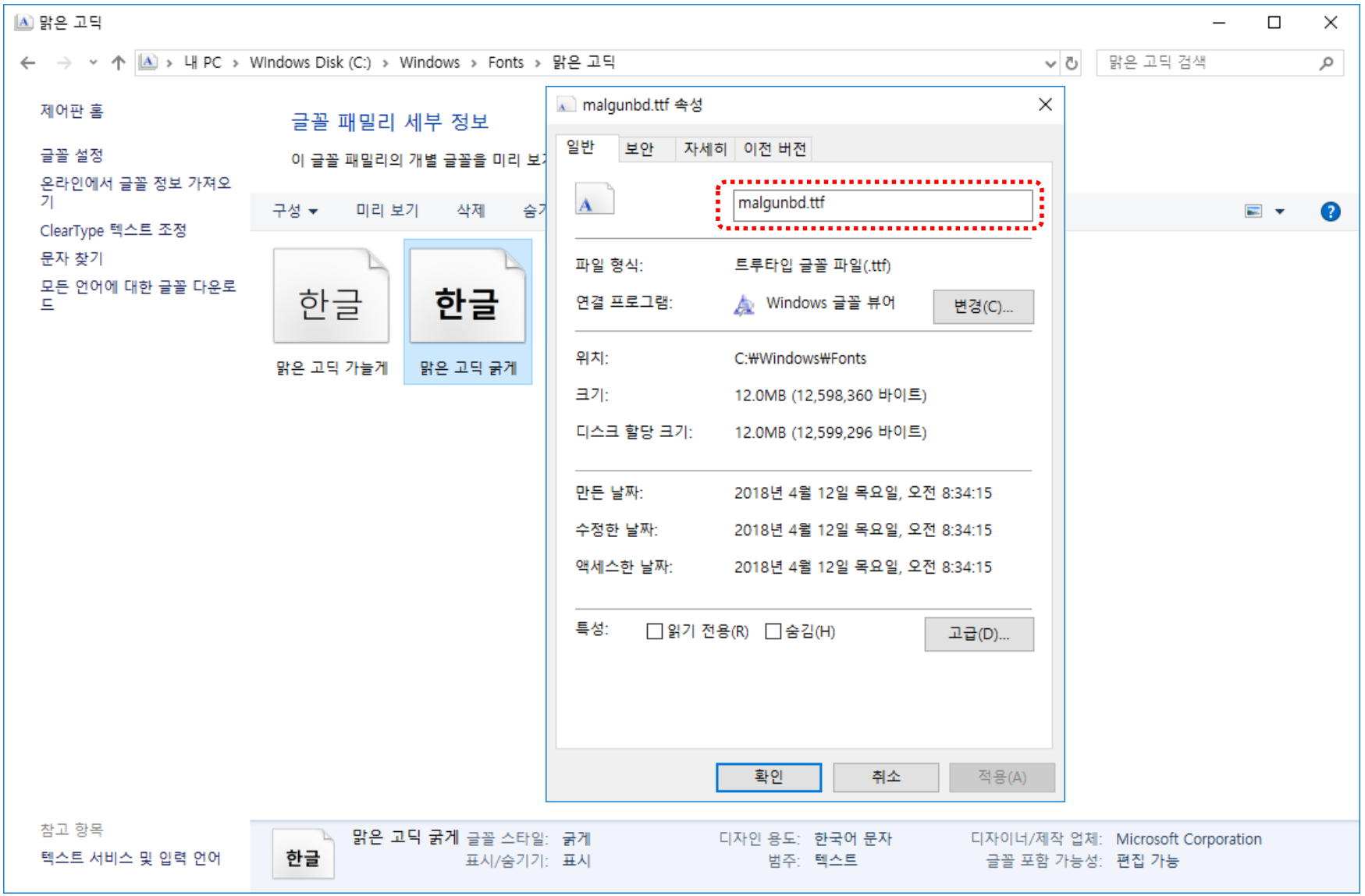
Out[8]: <function matplotlib.pyplot.show(*args, **kw)>



한글처리


Fonts
내 PC › Windows Disk (C:) › Windows › Fonts ›
Fonts 검색
제어판 홈
글꼴 설정
온라인에서 글꼴 정보 가져오기
ClearType 텍스트 조정
문자 찾기
모든 언어에 대한 글꼴 다운로드
컴퓨터에 설치되어 있는 글꼴을 미리 보거나 삭제하고 표시하거나 숨깁니다.
구성
이름 글꼴 스타일 표시/숨기기 디자인 용도 범주 디자이너/제작 업체
궁서체 보통 보통 표시 한국어 문자 텍스트 Han Yang I & C Co. Ltd.
돋움 보통 보통 표시 한국어 문자 텍스트 Han Yang I & C Co. Ltd.
돋움체 보통 보통 표시 한국어 문자 텍스트 Han Yang I & C Co. Ltd.
맑은 고딕 보통; 가늘게; 굵게 표시 한국어 문자 텍스트 Microsoft Corporation
문체부 궁체 정자체 보통 보통 표시
문체부 궁체 흘림체 보통 보통 표시
문체부 돋음체 보통 보통 표시
문체부 바탕체 보통 보통 표시
문체부 쓰기 정체 보통 보통 표시
문체부 쓰기 흘림체 보통 보통 표시
문체부 제목 돋음체 보통 보통 표시
문체부 제목 바탕체 보통 보통 표시
문체부 훈민정음체 보통 보통 표시
바탕 보통 보통 표시 한국어 문자 텍스트 Han Yang I & C Co. Ltd.
바탕체 보통 보통 표시 한국어 문자 텍스트 Han Yang I & C Co. Ltd.
새굴림 보통 보통 표시 한국어 문자 텍스트 HanYang System Co., LTD.
안상수2006가는 보통 보통 표시
안상수2006굵은 보통 보통 표시
안상수2006중간 보통 보통 표시
342개 항목
참고 항목
텍스트 서비스 및 입력 언어

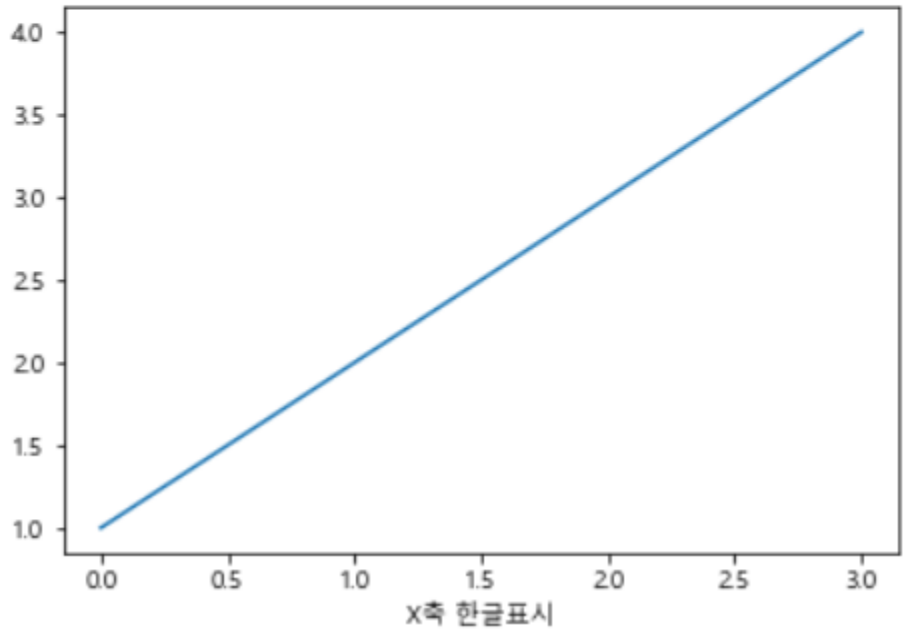
한글처리


맑은 고딕
내 PC › Windows Disk (C:) › Windows › Fonts › 맑은 고딕
맑은 고딕 검색
제어판 홈
글꼴 설정
온라인에서 글꼴 정보 가져오기
ClearType 텍스트 조정
문자 찾기
모든 언어에 대한 글꼴 다운로드
글꼴 패밀리 세부 정보
구성
미리 보기
삭제
한글
맑은 고딕 가늘게
한글
맑은 고딕 굵게
malgunbd.ttf 속성
일반
보안
자세히
이전 버전
malgunbd.ttf
파일 형식: 트루타입 글꼴 파일(.ttf)
연결 프로그램: Windows 글꼴 뷰어
변경(C)...
위치: C:₩Windows₩Fonts
크기: 12.0MB (12,598,360 바이트)
디스크 할당 크기: 12.0MB (12,599,296 바이트)
만든 날짜: 2018년 4월 12일 목요일, 오전 8:34:15
수정한 날짜: 2018년 4월 12일 목요일, 오전 8:34:15
액세스한 날짜: 2018년 4월 12일 목요일, 오전 8:34:15
특성: 읽기 전용(R) 숨김(H)
고급(D)...
확인
취소
적용(A)
참고 항목
텍스트 서비스 및 입력 언어
한글
맑은 고딕 굵게 글꼴 스타일: 굵게
표시/숨기기: 표시
디자인 용도: 한국어 문자
범주: 텍스트
디자이너/제작 업체: Microsoft Corporation
글꼴 포함 가능성: 편집 가능

한글처리

In [10]:

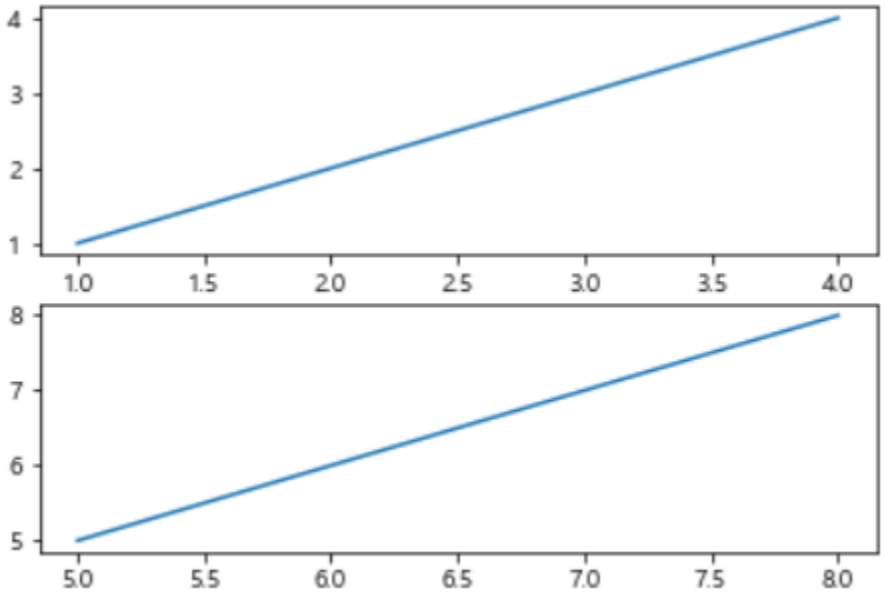
```
1 from matplotlib import font_manager, rc
2 import matplotlib
3 font_location="c:/Windows/fonts/malgunbd.ttf"
4 font_name = font_manager.FontProperties(fname=font_location).get_name()
5 matplotlib.rc('font', family=font_name)
6 plt.plot([1,2,3,4])
7 plt.xlabel("X축 한글표시")
8 plt.show()
```



여러 개의 그래프 그리기

In [15]:

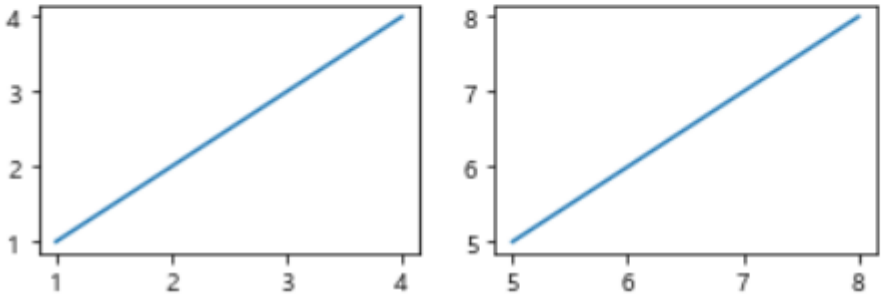
```
plt.figure()  # 하나의 캔버스를 생성
# subplot(m,n,idx)
# 매트릭스 형태로 행2 열1개인 창을 의미. idx는 mn형태의 idx번째
plt.subplot(2,1,1)
plt.plot([1,2,3,4],[1,2,3,4])
plt.subplot(2,1,2)
plt.plot([5,6,7,8],[5,6,7,8])
plt.show()
```



여러 개의 그래프 그리기

In [18]:

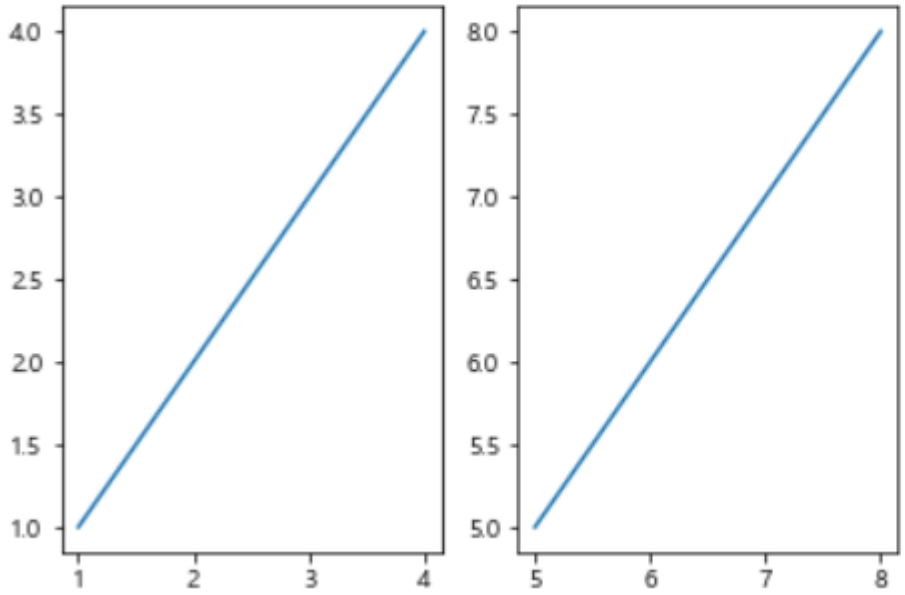
```
1 plt.figure()
2 plt.subplot(2,2,1)
3 plt.plot([1,2,3,4],[1,2,3,4])
4 plt.subplot(2,2,2)
5 plt.plot([5,6,7,8],[5,6,7,8])
6 plt.show()
```



여러 개의 그래프 그리기

In [19]:

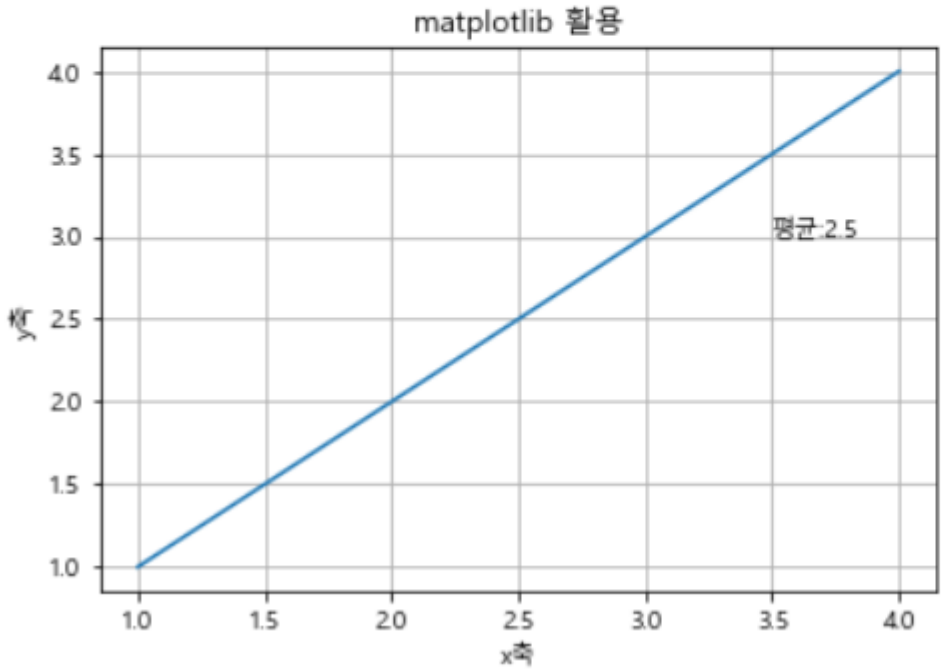
```
1 plt.figure()
2 plt.subplot(1,2,1) # 1행의 첫 번째 컬럼
3 plt.plot([1,2,3,4],[1,2,3,4])
4 plt.subplot(1,2,2) # 1행의 두 번째 컬럼
5 plt.plot([5,6,7,8],[5,6,7,8])
6 plt.show()
```



그래프에 문자 삽입

In [22]:

```
1 plt.plot([1,2,3,4],[1,2,3,4])
2 plt.xlabel('x축')
3 plt.ylabel('y축')
4 plt.title('matplotlib 활용')
5 plt.text(3.5, 3.0, '평균:2.5')
6 plt.grid(True)
7 plt.show()
```



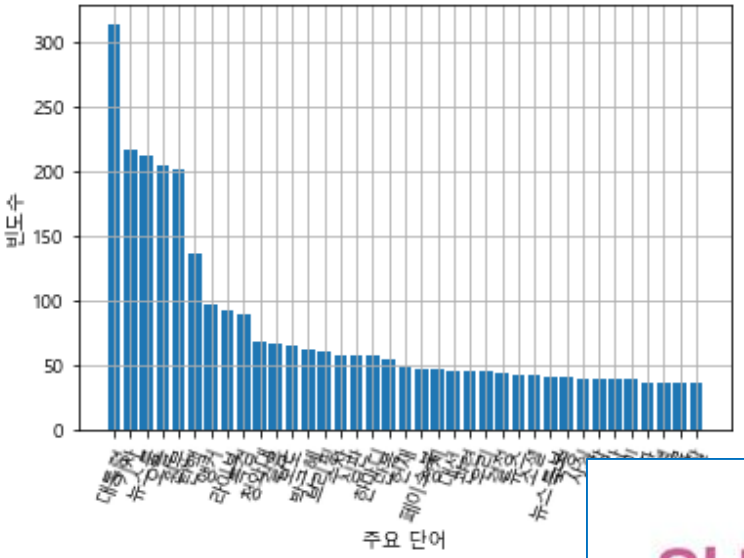
Matplotlib

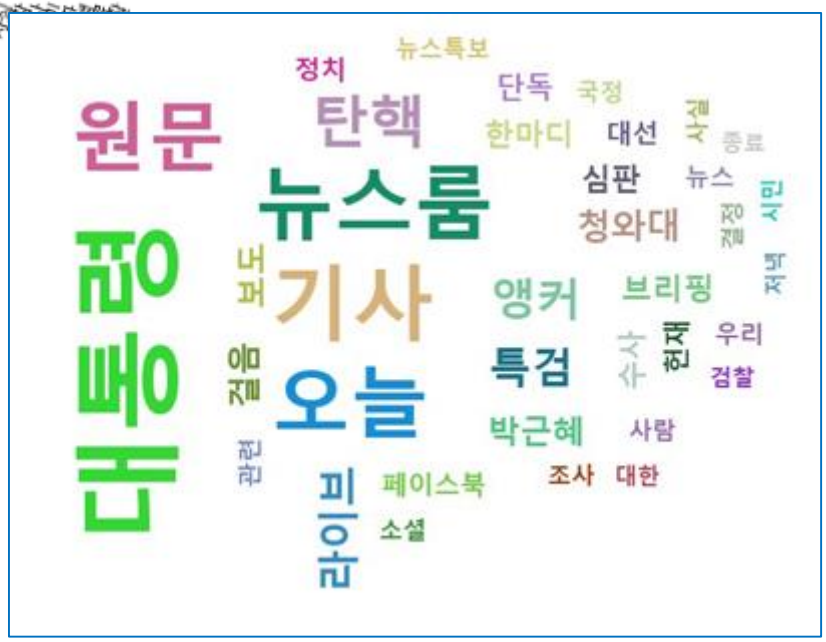
빈도분석

k-최근접 이웃 알고리즘(k-Nearest Neighbors)

명사 추출 및 빈도 분석

대통령 : 314
기사 : 217
뉴스룸 : 212
오늘 : 204
원문 : 202
탄핵 : 137
앵커 : 97
라이브 : 92
특검 : 90
청와대 : 69
걸음 : 67
보도 : 65
박근혜 : 62
브리핑 : 61
수사 : 58
심판 : 58
한마디 : 57
단독 : 54
헌재 : 49
페이스북 : 47
정치 : 47
대선 : 46
관련 : 45
우리 : 45
결정 : 44
뉴스 : 43
소셜 : 42
뉴스특보 : 41
국정 : 41
사실 : 40
사람 : 40
조사 : 40
시민 : 39





명사 추출 및 빈도 분석

- 페이스북의 JTBC뉴스와 조선일보데이터 분석
  - 기간 : 2016-10-01~2017-03-12
- 설치 패키지
  - pip install JPype1
  - pip install KoNLPy
  - pip install pytagcloud
  - pip install pygame
  - pip install simplejson

명사 추출 및 빈도 분석

In [3]:

```
import json
import re

from konlpy.tag import Okt
from collections import Counter

import matplotlib.pyplot as plt
import matplotlib
from matplotlib import font_manager, rc

import pytagcloud
import webbrowser

#[CODE 1]
def showGraph(wordInfo):

    font_location = "c:/Windows/fonts/malgunbd.ttf"
    font_name = font_manager.FontProperties(fname=font_location).get_name()
    matplotlib.rc('font', family=font_name)

    plt.xlabel('주요 단어')
    plt.ylabel('빈도수')
    plt.grid(True)
    # 최대빈도수값과 최대빈도수 단어가 저장
    Sorted_Dict_Values = sorted(wordInfo.values(), reverse=True)
    Sorted_Dict_Keys = sorted(wordInfo, key=wordInfo.get, reverse=True)
    # 막대그래프 그리는 함수
    plt.bar(range(len(wordInfo)), Sorted_Dict_Values, align='center')
    # x축의 각 데이터 별 문자열(tick)을 지정
    plt.xticks(range(len(wordInfo)), list(Sorted_Dict_Keys), rotation='70')

    plt.show()
```

명사 추출 및 빈도 분석

In [4]:

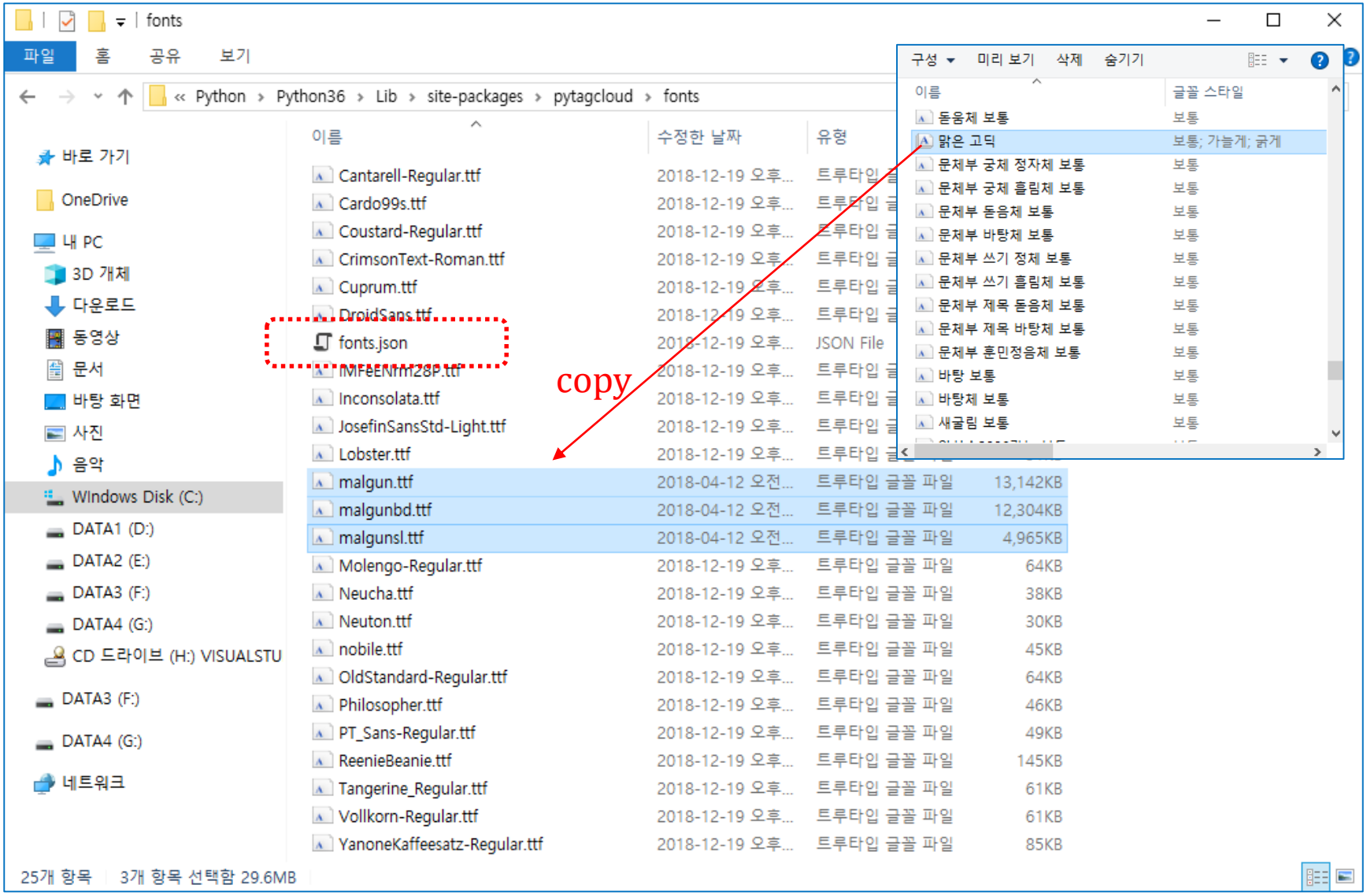
```
#[CODE 2]
def saveWordCloud(wordInfo, filename):

    taglist = pytagcloud.make_tags(dict(wordInfo).items(), maxsize=80)
    pytagcloud.create_tag_image(taglist, filename, size=(640, 480), fontname='korean', rectangular=False)
    webbrowser.open(filename)

```

명사 추출 및 빈도 분석

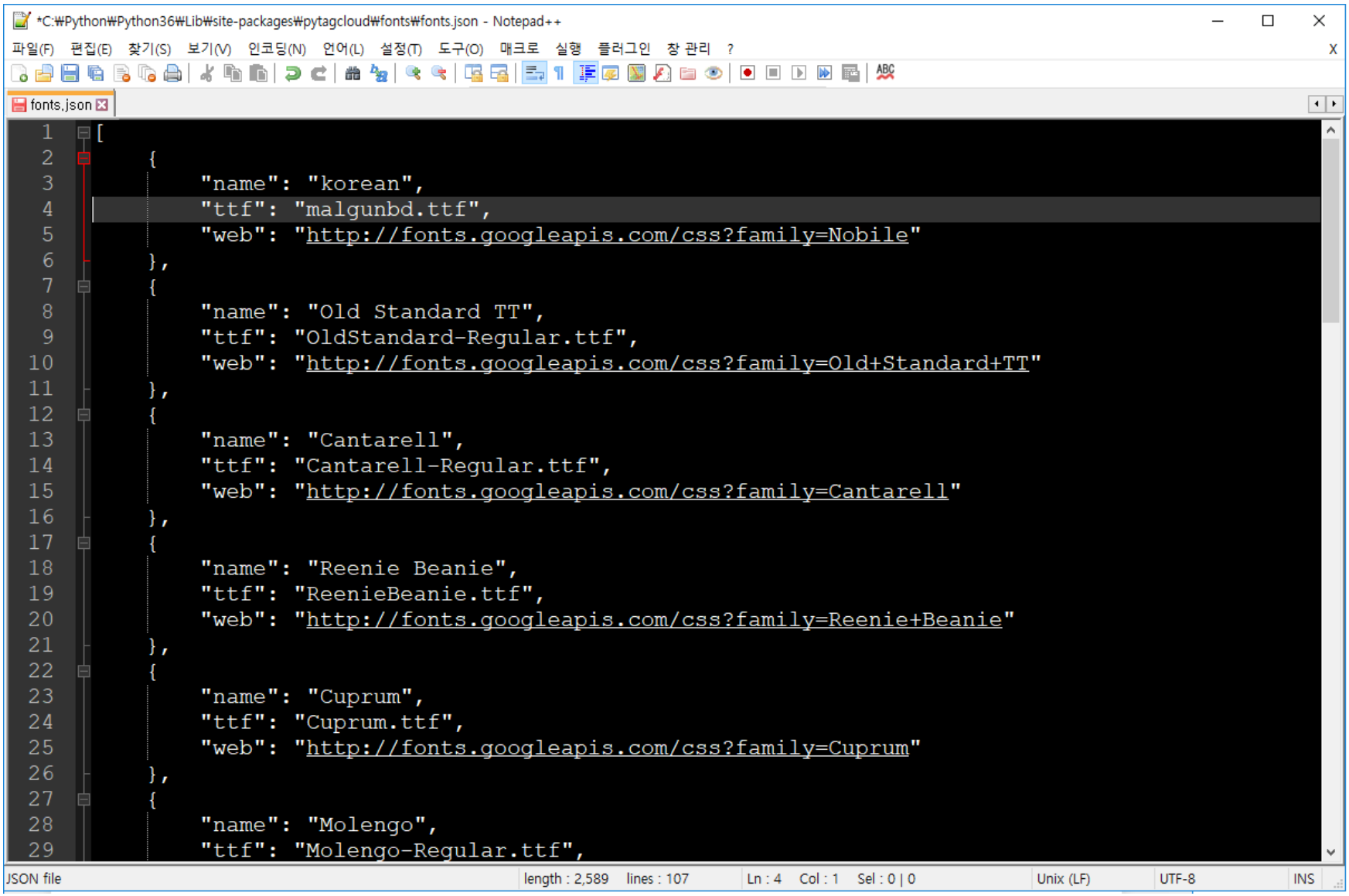
In [11]:

```
def main():

    openFileName = 'G:₩python_workspace₩jupyter₩BigData/jtbcnews_facebook_2016-10-01_2017-03-12.json'
    #openFileName = 'G:/python_workspace/jupyter/BigData/chosun_facebook_2016-10-01_2017-03-12.json'

    cloudImagePath = openFileName + '.jpg'

    rfile = open(openFileName, 'r', encoding='utf-8').read()

    jsonData = json.loads(rfile)
    message = ''

    # jsonData의 개별 message를 합쳐서 하나의 문자열로 구성, 불필요한 ₩t, ₩n등의 문자 제거
    for item in jsonData:
        if 'message' in item.keys():
            message = message + re.sub(r'[^₩w]', ' ', item['message']) + ' '

    # 품사 클래스 - 명사만 추출하여 갯수를 세어 상위 50개만 가지고 온다.
    nlp = Okt()
    nouns = nlp.nouns(message)
    count = Counter(nouns)

    wordInfo = dict()
    for tags, counts in count.most_common(50):
        if (len(str(tags)) > 1):
            wordInfo[tags] = counts
            print ("%s : %d" % (tags, counts))

    showGraph(wordInfo)
    saveWordCloud(wordInfo, cloudImagePath)

if __name__ == "__main__":
    main()
```

한글폰트 지정


fonts
파일
홈
공유
보기
« Python › Python36 › Lib › site-packages › pytagcloud › fonts
바로 가기
OneDrive
내 PC
3D 개체
다운로드
동영상
문서
바탕 화면
사진
음악
Windows Disk (C:)
DATA1 (D:)
DATA2 (E:)
DATA3 (F:)
DATA4 (G:)
CD 드라이브 (H:) VISUALSTU
DATA3 (F:)
DATA4 (G:)
네트워크
이름
수정한 날짜
유형
Cantarell-Regular.ttf
Cardo99s.ttf
Coustard-Regular.ttf
CrimsonText-Roman.ttf
Cuprum.ttf
DroidSans.ttf
fonts.json
JSON File
Inconsolata.ttf
JosefinSansStd-Light.ttf
Lobster.ttf
malgun.ttf
2018-04-12 오전...
트루타입 글꼴 파일
13,142KB
malgunbd.ttf
12,304KB
malgunsl.ttf
4,965KB
Molengo-Regular.ttf
64KB
Neucha.ttf
38KB
Neuton.ttf
30KB
nobile.ttf
45KB
OldStandard-Regular.ttf
64KB
Philosopher.ttf
46KB
PT_Sans-Regular.ttf
49KB
ReenieBeanie.ttf
145KB
Tangerine_Regular.ttf
61KB
Vollkorn-Regular.ttf
61KB
YanoneKaffeesatz-Regular.ttf
85KB
2018-12-19 오후...
25개 항목
3개 항목 선택함 29.6MB
copy
구성
미리 보기
삭제
숨기기
이름
글꼴 스타일
돋움체 보통
맑은 고딕
보통; 가늘게; 굵게
문체부 궁체 정자체 보통
문체부 궁체 흘림체 보통
문체부 돋움체 보통
문체부 바탕체 보통
문체부 쓰기 정체 보통
문체부 쓰기 흘림체 보통
문체부 제목 돋움체 보통
문체부 제목 바탕체 보통
문체부 훈민정음체 보통
바탕 보통
바탕체 보통
새굴림 보통

한글폰트 지정

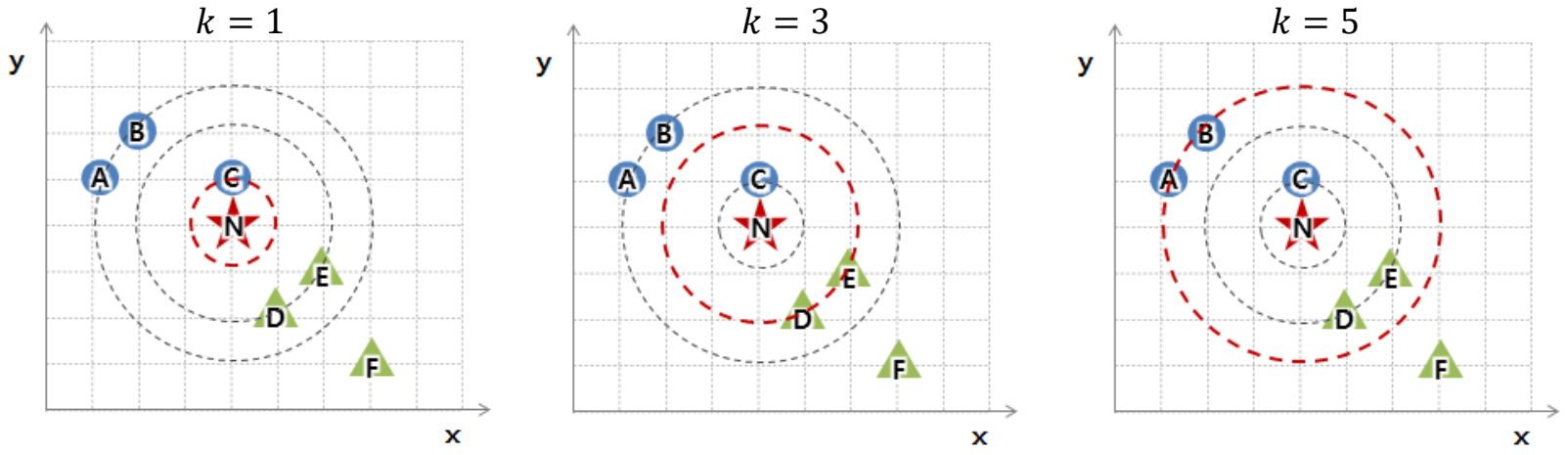

*C:₩Python₩Python36₩Lib₩site-packages₩pytagcloud₩fonts₩fonts.json - Notepad++
파일(F) 편집(E) 찾기(S) 보기(V) 인코딩(N) 언어(L) 설정(T) 도구(O) 매크로 실행 플러그인 창 관리 ?
fonts.json
[
{
"name": "korean",
"ttf": "malgunbd.ttf",
"web": "http://fonts.googleapis.com/css?family=Nobile"
},
{
"name": "Old Standard TT",
"ttf": "OldStandard-Regular.ttf",
"web": "http://fonts.googleapis.com/css?family=Old+Standard+TT"
},
{
"name": "Cantarell",
"ttf": "Cantarell-Regular.ttf",
"web": "http://fonts.googleapis.com/css?family=Cantarell"
},
{
"name": "Reenie Beanie",
"ttf": "ReenieBeanie.ttf",
"web": "http://fonts.googleapis.com/css?family=Reenie+Beanie"
},
{
"name": "Cuprum",
"ttf": "Cuprum.ttf",
"web": "http://fonts.googleapis.com/css?family=Cuprum"
},
{
"name": "Molengo",
"ttf": "Molengo-Regular.ttf",
JSON file
length : 2,589 lines : 107
Ln : 4 Col : 1 Sel : 0 | 0
Unix (LF)
UTF-8
INS

Matplotlib
빈도분석
k-최근접 이웃 알고리즘(k-Nearest Neighbors)

개요

- k-NN($k-Nearest\ Neighbors$)은 머신러닝 분야에서 가장 간단한 알고리즘
- 훈련 데이터셋을 그냥 저장하는 것이 모델을 만드는 과정의 전부
- 새로운 데이터 포인트에 대해 예측할 땐 훈련 데이터셋에서 가장 가까운 데이터 포인트(최근접 이웃)을 찾는다.

- k-NN은 새로 들어온 "★은 ○ (or △) 그룹의 데이터와 가장 가까우니 ★은 ○ (or △) 그룹이다." 라고 분류하는 알고리즘이다.



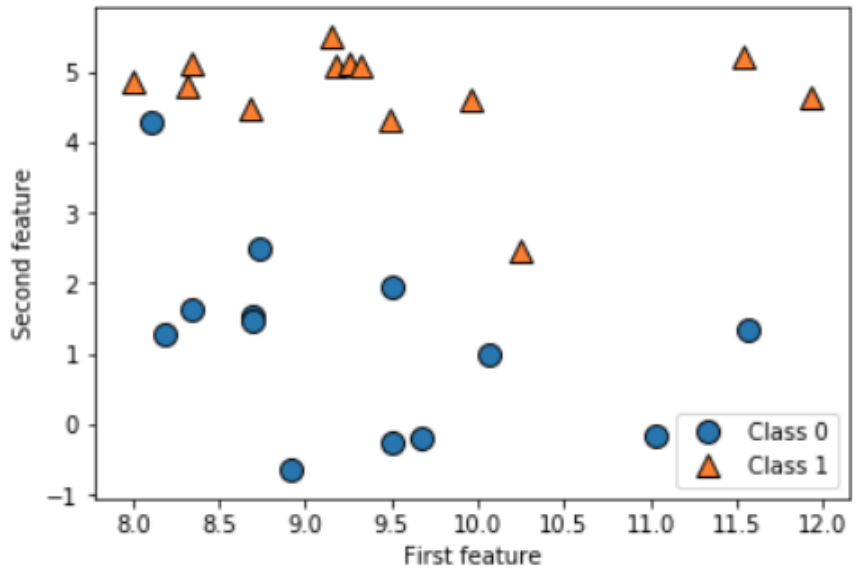
https://kkokkilkon.tistory.com/14

예제에 사용할 데이터 셋

- k-최근접 이웃 알고리즘(k-Nearest Neighbors Algorithm)

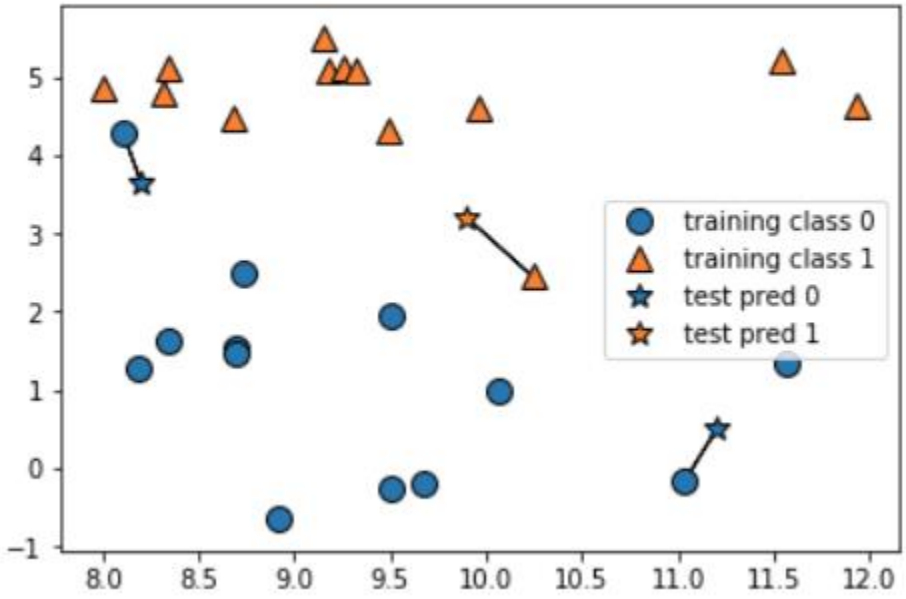
```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import mglearn as mglearn
4
5  # generate dataset
6  X, y = mglearn.datasets.make_forge()
7  # plot dataset
8  mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
9  plt.legend(["Class 0", "Class 1"], loc=4)
10 plt.xlabel("First feature")
11 plt.ylabel("Second feature")
12 print("X.shape:", X.shape)
```

```
X.shape: (26, 2)
```



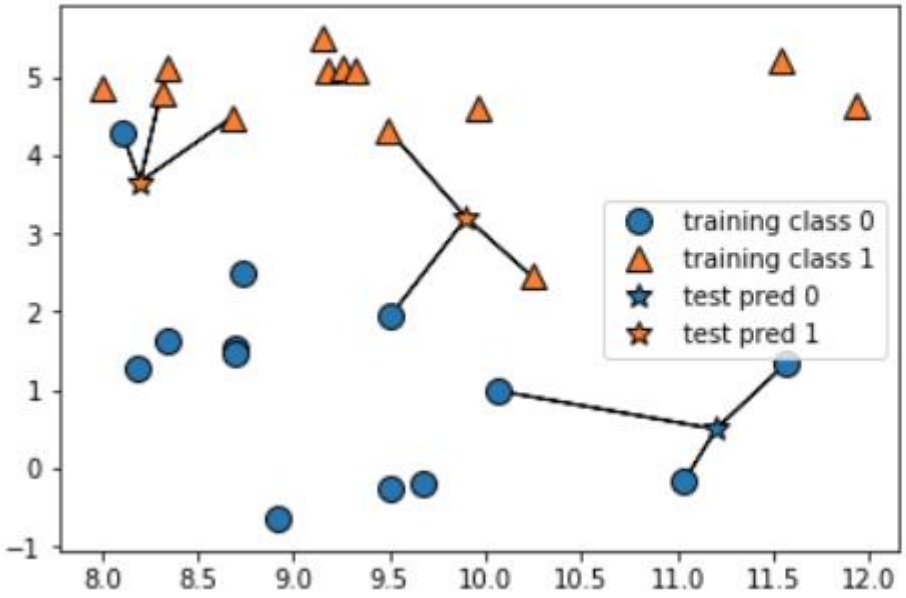
- 가장 가까운 훈련 데이터 포인트 하나를 최근접 이웃으로 찾아 예측에 사용
- 데이터 포인터 3개 추가 : 1-최근접 이웃 알고리즘 예측

```
1  mglearn.plots.plot_knn_classification(n_neighbors=1)
```



- 3-최근접 이웃 알고리즘 예측

```
1 mglearn.plots.plot_knn_classification(n_neighbors=3)
```



scikit-learn을 이용한 k-NN 알고리즘 적용

- k-최근접 이웃 알고리즘(k-Nearest Neighbors Algorithm)

```
from sklearn.model_selection import train_test_split
X, y = mglearn.datasets.make_forge()

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors=3)  # 이웃의 수 : 3

# 훈련 세트를 사용하여 분류 모델 학습
clf.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=3, p=2,
           weights='uniform')
```

```
# 테스트 데이터에 대해 predict 메서드를 호출해서 예측
print("Test set predictions:", clf.predict(X_test))
```

```
Test set predictions: [1 0 1 0 1 0 0]
```

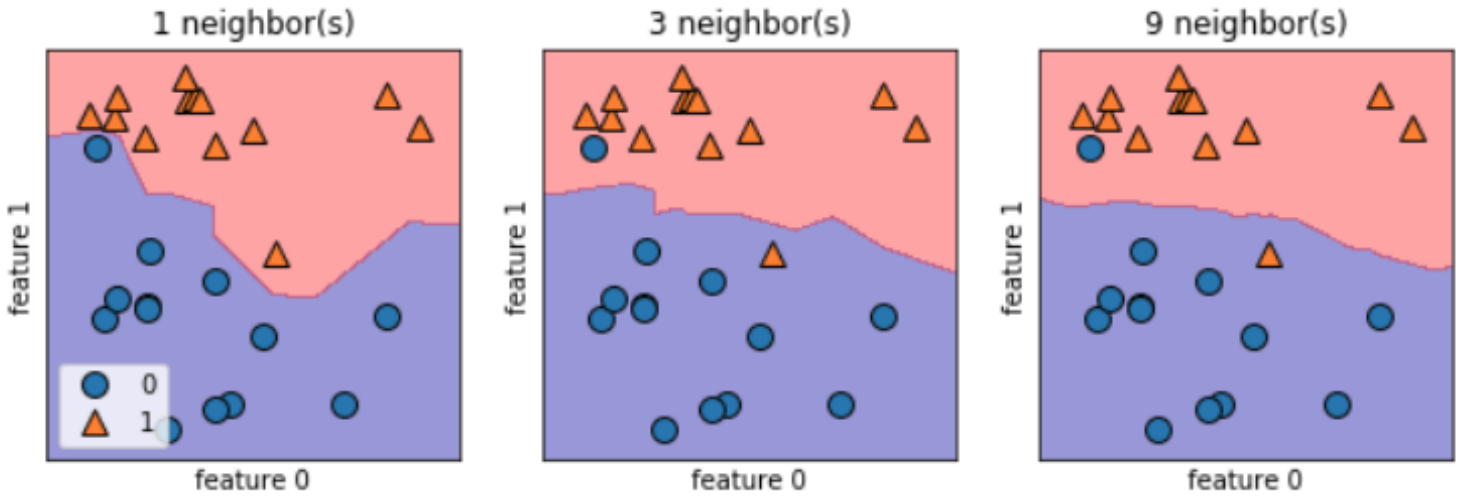
```
# 모델이 얼마나 잘 일반화되었는지 평가
# 테스트 데이터와 테스트 레이블을 넣어 호출
print("Test set accuracy: {:.2f}".format(clf.score(X_test, y_test)))
```

```
Test set accuracy: 0.86
```

KNeighborsClassifier 분석


```
fig, axes = plt.subplots(1, 3, figsize=(10, 3))

for n_neighbors, ax in zip([1, 3, 9], axes):
    # the fit method returns the object self, so we can instantiate
    # and fit in one line
    clf = KNeighborsClassifier(n_neighbors=n_neighbors).fit(X, y)
    mglearn.plots.plot_2d_separator(clf, X, fill=True, eps=0.5, ax=ax, alpha=.4)
    mglearn.discrete_scatter(X[:, 0], X[:, 1], y, ax=ax)
    ax.set_title("{} neighbor(s)".format(n_neighbors))
    ax.set_xlabel("feature 0")
    ax.set_ylabel("feature 1")
axes[0].legend(loc=3)
```

```
<matplotlib.legend.Legend at 0x117bf1ef160>
```



유방암 데이터셋 사용

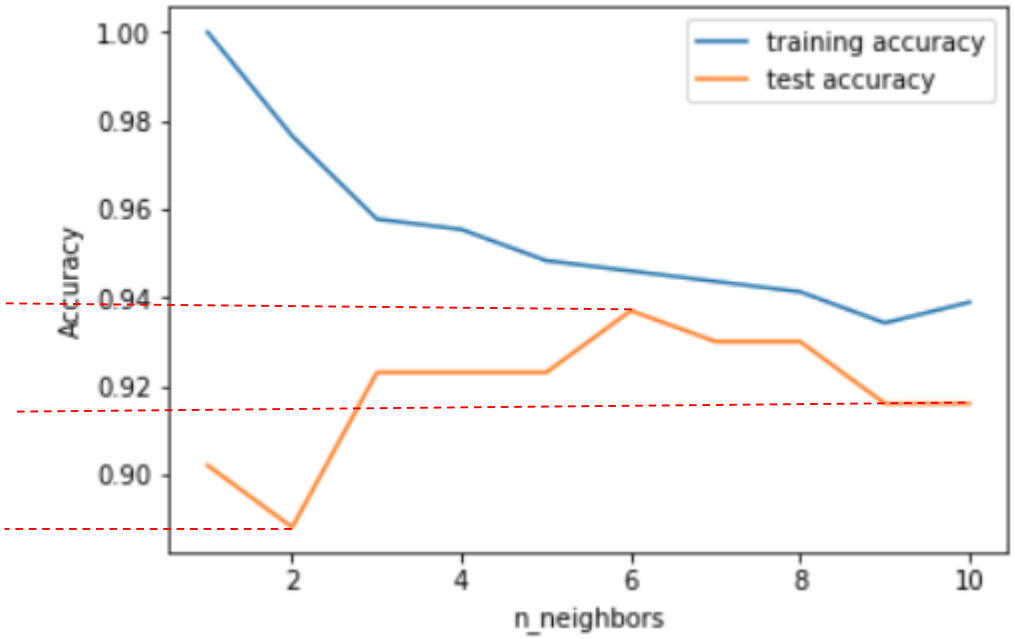
- 먼저 훈련 세트와 테스트 세트로 나눈 후 이웃의 수를 달리하여 훈련 세트와 테스트 세트의 성능 평가

In [34]:

```
from sklearn.datasets import load_breast_cancer

cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, stratify=cancer.target, random_state=66)

training_accuracy = []
test_accuracy = []
# try n_neighbors from 1 to 10
neighbors_settings = range(1, 11)

for n_neighbors in neighbors_settings:
    # build the model
    clf = KNeighborsClassifier(n_neighbors=n_neighbors)
    clf.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(clf.score(X_train, y_train))
    # record generalization accuracy
    test_accuracy.append(clf.score(X_test, y_test))

plt.plot(neighbors_settings, training_accuracy, label="training accuracy")
plt.plot(neighbors_settings, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbors")
plt.legend()
```

유방암 데이터셋 사용

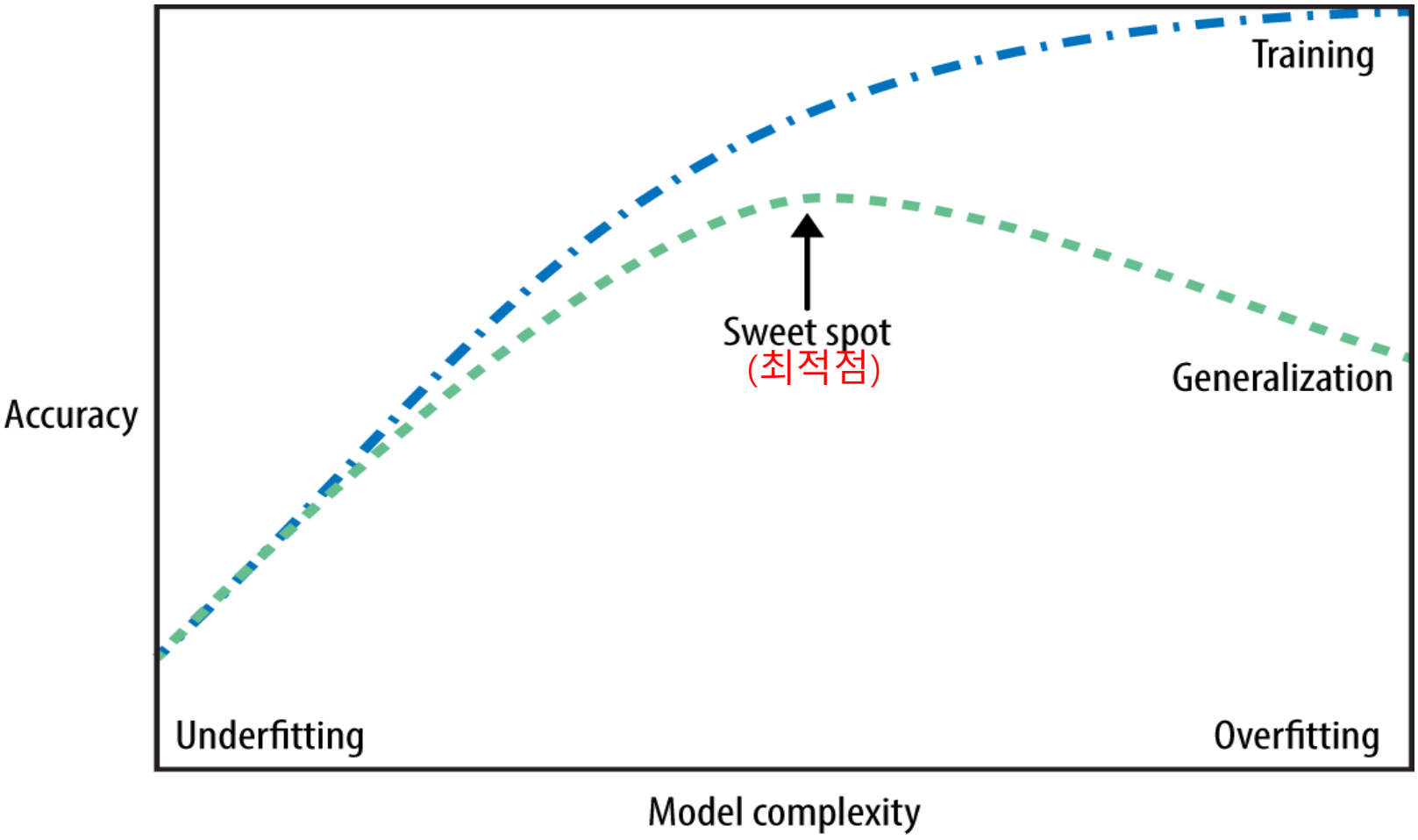

```
Out[34]: <matplotlib.legend.Legend at 0x117bf2ced68>
```



- 이웃을 하나 사용한 테스트 세트의 정확도는 이웃을 많이 사용했을 때보다 낮다.
- 그러나 10개 사용했을 때는 모델이 너무 단순해서 정확도는 더 나빠진다.
- 가장 좋을 때는 중간 정도인 여섯 개를 사용한 경우이다.
- 가장 나쁜 정확도도 88%이므로 수긍할 만하다.

모델 복잡도에 따른 훈련과 테스트 정확도의 변화

- k-최근접 이웃 알고리즘(k-Nearest Neighbors Algorithm)



장단점

- 장점
  - 이해하기 매우 쉬운 모델
  - 많이 조정하지 않아도 자주 좋은 성능 발휘
  - 더 복잡한 알고리즘을 적용해 보기 전에 시도해 볼 수 있는 좋은 시작점
- 단점
  - 훈련세트가 매우 크면 예측이 느려진다.
  - (수백 개 이상의)많은 특성을 가진 데이터 셋에는 잘 동작하지 않으며,
  - 대부분이 0인(희소)데이터 셋은 특히 잘 작동되지 않는다.
  - 예측이 느리고 많은 특성을 처리하는 능력이 부족해 잘 사용하지 않는다.