빅데이터분석 실습

한글 텍스트 분석

데이터 사이언스 전공

담당교수: 곽철완

강의 내용

- KoNLPy 설치
- 네이버 영화 리뷰 분석
- 제약회사 데이터 분석

준비 작업

▪ KoNLPy 설치

- 버전 확인
  - 윈도우 버전 64비트
  - 파이썬 버전 64비트 일치해야 함
  - 자바 버전 1.7 이상
    - 윈도우 시작 버튼을 눌러서 Java 폴더를 찾음
    - Java 폴더를 누르고 Java 정보를 눌러 버전 확인

- 64비트 OS에는 win-amd64 파일 사용
- KoNLPy 설치 방법 팁
  - https://hogni.tistory.com/39

- 기본 라이브러리 import

```
%matplotlib inline
from IPython.display import display
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import mglearn
import sklearn
```

1. 네이버 영화 리뷰 분석

▪ 데이터 세트 불러오기

- 네이버 영화 리뷰 데이터 세트(https://github.com/e9t/nsmc)
  - rating_train.txt와 rating_test.txt 데이터 세트를 다운받아 작업 폴더에 저장한다
- pandas를 이용하여 데이터 세트를 불러들인다
  - 데이터 항목은 id, document, label 3개이고 탭으로 구분되어 있다
  - pandas의 read_csv 메서드를 사용할 때 데이터 세트가 탭으로 구분되어 있으므로 구분자를 탭(delimiter = '\t')으로 지정한다
  - 데이터 세트에 빈 문자열이 있어도 nan으로 저장되지 않도록 'keep_default_na = False'로 지정하여, 빈 문자열이 그대로 저장되게 한다
- 데이터를 읽는다

```
df_train = pd.read_csv(“E:/workspace/Korean/rating_train.txt”, delimiter = ‘\t’,
                       keep_default_na=False)
df_train.head(n=3)
```

```
df_train = pd.read_csv("E:/workspace/korean/ratings_train.txt", delimiter = '\t',
                       keep_default_na=False)
df_train.head(n=3)
```

	id	document	label
0	9976970	아 더빙.. 진짜 짜증나네요 목소리	0
1	3819312	흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나	1
2	10265843	너무재밋었다그래서보는것을추천한다	0

- 사용할 데이터는 document와 label이다
- label이 0이면 부정 리뷰이고, 1이면 긍정이다

- pandas 데이터 프레임을 NumPy 배열로 바꾸어준다
- 평가용 데이터도 동일하게 NumPy 배열로 바꾸어준다

```
text_train, y_train = df_train[‘document’].values, df_train[‘label’].values

df_test = pd.read_csv(“E:/workspace/Korean/rating_test.txt”, delimiter = ‘\t’,
                      keep_default_na=False)
text_test, y_test = df_train[‘document’].values, df_train[‘label’].values
```

```
text_train, y_train = df_train['document'].values, df_train['label'].values

df_test = pd.read_csv("E:/workspace/korean/ratings_test.txt", delimiter = '\t',
                      keep_default_na=False)
text_test, y_test = df_test['document'].values, df_test['label'].values
```

◦ 학습용 데이터 세트와 평가용 데이터 세트 크기와 클래스 비율 확인

```
len(text_train), np.bincount(y_train)

len(text_test), np.bincount(y_test)
```

```
len(text_train), np.bincount(y_train)
(150000, array([75173, 74827], dtype=int64))

len(text_test), np.bincount(y_test)
(50000, array([24827, 25173], dtype=int64))
```

◦ 클래스를 확인한 결과 양성과 음성의 숫자가 비슷하다

▪ KoNLPy의 Okt

◦ Okt 클래스 객체를 만든다

```
from konlpy.tag import Okt
okt_tag = Okt( )
```

```
from konlpy.tag import Okt
okt_tag = Okt()
```

◦ TfidfVectorizer의 tokenizer 매개변수를 삽입할 함수를 만든다

◦ 이 함수는 텍스트 하나를 받아서 Okt의 형태소 분석 메서드인 morphs에 받은 문자열의 리스트를 그대로 반환한다

```
def okt_tokenizer(text):
        return okt_tag.morphs(text)
```

```
def okt_tokenizer(text):
    return okt_tag.morphs(text)
```

▪ 매개변수를 위한 그리드 서치

- TfidfVectorizer의 min_df 와 ngram_range
- LogisticRegression의 규제 매개변수 C에 대한 그리드 서치 적용

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV

okt_param_grid = {'tfidfvectorizer__min_df' : [3, 5, 7],
                  'tfidfvectorizer__ngram_range' : [(1, 1), (1, 2), (1, 3)],
                  'logisticregression__C' : [0.1, 1, 10]}
okt_pipe = make_pipeline(TfidfVectorizer(tokenizer=okt_tokenizer),
                         LogisticRegression(solver = 'liblinear'))
```

뒷장에 계속

```
okt_grid = GridSearchCV(okt_pipe, okt_param_grid, cv=3)

okt_grid.fit(text_train[0:2000], y_train[0:2000])
print("최상의 교차 검증 점수: {:.2f}".format(okt_grid.best_score_))
print("최적의 교차 검증 매개변수: ", okt_grid.best_params_)
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV

okt_param_grid = {'tfidfvectorizer__min_df' : [3, 5, 7],
                  'tfidfvectorizer__ngram_range' : [(1, 1), (1, 2), (1, 3)],
                  'logisticregression__C' : [0.1, 1, 10]}
okt_pipe = make_pipeline(TfidfVectorizer(tokenizer=okt_tokenizer),
                                         LogisticRegression(solver = 'liblinear'))
okt_grid = GridSearchCV(okt_pipe, okt_param_grid, cv=3)

okt_grid.fit(text_train[0:2000], y_train[0:2000])
print("최상의 교차 검증 점수: {:.2f}".format(okt_grid.best_score_))
print("최적의 교차 검증 매개변수:  ", okt_grid.best_params_)
```

```
최상의 교차 검증 점수: 0.73
최적의 교차 검증 매개변수:  {'logisticregression__C': 1, 'tfidfvectorizer__min_df':
3, 'tfidfvectorizer__ngram_range': (1, 1)}
```

- 한참 후(약 5분 이상), 데이터 2,000개를 사용하여 73%의 교차 검증 점수를 얻었다

```
X_test_okt = okt_grid.best_estimator_.named_steps[“tfidfvectorizer”].
                    transform(text_test)
score = okt_grid.best_estimator_.named_steps[“logisticregression”].
                    score(X_test_okt, y_test)
print(“평가용 세트 점수: {:.2f}”.format(score))
```

```
X_test_okt = okt_grid.best_estimator_.named_steps["tfidfvectorizer"].transform(text_test)
score = okt_grid.best_estimator_.named_steps["logisticregression"].score(X_test_okt, y_test)
print("평가용 세트 점수: {:.2f}".format(score))
```

```
평가용 세트 점수: 0.75
```

◦ 평가용 데이터의 점수 확인은 파이프 라인의 ‘tfidfvectorizer’ step에서 transform 메서드를 호출한 후, 변환된 데이터를 이용하여 ‘logisticregression’ step의 score 함수를 호출한다

```
okt_pipe = make_pipeline(TfidfVectorizer(tokenizer=okt_tokenizer),
                                        LogisticRegression(solver = 'liblinear'))
okt_grid = GridSearchCV(okt_pipe, okt_param_grid, cv=3)
```

파이프 라인으로 만든 함수

```
X_test_okt = okt_grid.best_estimator_.named_steps["tfidfvectorizer"].transform(text_test)
score = okt_grid.best_estimator_.named_steps["logisticregression"].score(X_test_okt, y_test)
```

2. 제약회사 데이터 분석

▪ 데이터 세트

- <출처> '파이썬을 이용한 빅데이터 분석'(2017) 유성준, Kmooc
- 국내 제약회사의 제품 체험단 신청 고객들의 데이터 2,874건
- 파일 이름: customer_data.csv
- 피처
  - Review : 신청글
  - SNS : 체험 내용 SNS 경로
  - Addr : 주소
  - Score: 마케팅 담당자가 고객 체험단 활동을 평가 후 부여한 점수 (1 ~ 5)

- 파일 불러오기
  - cp949 한글을 위한 문자 인코딩

```
customer = pd.read_csv("E:/workspace/customer_data.csv", encoding = "cp949")
customer.head(n=5)
```

	Unnamed: 0	Score	Review	SNS	Addr
0	0	5	안녕하세요. 데일리 신청해 봅니다. 어렸을 적 부터 장이 좋지 않았는데 고기 인...	twitter	경기도
1	1	2	오!! 안그래도 장이 안좋아서 아침마다 고생이거든요~~ 먹고 건강해지고 싶네용 ^^	facebook	서울특별시
2	2	2	요즘 장이 안좋은지 하루종일 더부룩하고 배변후에도 시원하지 않네요.꼭 체험해보고싶습니다.	facebook	서울특별시

■ 피처값 추출

- 한국어 형태소 분석기 사용한 피처값 추출
  - KoNLPy에서 제공하는 한국어 형태소 분석기 사용
  - 한국어 형태소 분석기를 사용해 명사와 형용사를 추출
  - 문서 단어 행렬(document-term matrix)을 생성하기 위해 토큰 추출
  - 고객의 의지가 표현된 피처값을 추출하기 위해 tf-idf값을 이용
  - 형태소 분석기 Kkma, Komoran, Hannanum, Okt(이전 Twitter), Mecab 중 Okt 사용

▪ Okt import 및 선언

- Okt 형태소 분석기 import
- 형태소 분석기 Okt 생성

```
from konlpy.tag import Okt
okt_tag = Okt( )
```

```
from konlpy.tag import Okt
okt_tag = Okt()
```

- Okt 를 사용한 특정 품사 추출 함수 생성
  - Okt 형태소 분석기로 명사, 형용사를 추출하는 함수 tokenize 생성한다
  - for 문을 이용하여 토큰의 품사가 명사 혹은 형용사일 경우 리스트 변수 stem에 저장한다
  - pos 메서드는 텍스트를 토큰화하여 품사를 판별하고 태그를 붙여준다

```
def tokenize(test):
    stems = [ ]
    tagged = okt_tag.pos(text)
    for i in range(0, len(tagged))
          if (tagged[i][1] == 'Noun' or tagged[i][1] == 'Adjective'
              stems.append(tagged[i][0])
     return stems
```

```
def tokenize(text):
    stems = []
    tagged = okt_tag.pos(text)
    for i in range (0, len(tagged)):
        if (tagged[i][1] == 'Noun' or tagged[i][1] == 'Adjective'):
            stems.append(tagged[i][0])
    return stems
```

- Okt 형태소를 사용한 명사 추출
  - 고객의 신청글 항목의 0번째 텍스트를 형태소 분석
  - 토큰의 품사가 명사인 경우 출력
  - for 문을 빼고 tagged를 출력하면 모든 토큰의 품사를 확인할 수 있다

```
tagged = okt_tag.pos(customer['Review'][0])
for i in range (0, len(tagged)):
    if (tagged[i][1] == 'Noun'):
        print(tagged[i])
```

```
('데', 'Noun')
('일리', 'Noun')
('신청', 'Noun')
('적', 'Noun')
('부터', 'Noun')
('고기', 'Noun')
('인스턴트', 'Noun')
('음식', 'Noun')
('장', 'Noun')
('상태', 'Noun')
('더욱', 'Noun')
('업무', 'Noun')
('스트레스', 'Noun')
('또한', 'Noun')
('것', 'Noun')
```

```
tagged = okt_tag.pos(customer['Review'][0])
print(tagged)
```

```
[('안녕하세요', 'Adjective'), ('.', 'Punct
('해', 'Verb'), ('봅니다', 'Verb'), ('.',
n'), ('장이', 'Suffix'), ('좋지', 'Adjecti
('음식', 'Noun'), ('을', 'Josa'), ('좋아하
n'), ('가', 'Josa'), ('더욱', 'Noun'), ('
('업무', 'Noun'), ('에', 'Josa'), ('지쳐',
ve'), ('것', 'Noun'), ('같구요', 'Adjectiv
니다', 'Verb'), ('.', 'Punctuation'), ('빨
ective'), ('잘', 'Noun'), ('을', 'Josa'),
b'), ('^^', 'Punctuation')]
```

▪ 문서 단어 행렬 구성

- TfidfVectorizer 모듈 import → 텍스트에서 tf-idf 값을 구하기 위함
- 데이터 프레임의 텍스트를 astype 메서드를 사용하여 문자열 형태로 변환
- tolist 메서드를 사용하여 리스트로, array 메서드를 사용하여 배열로 순차적 변환

```
from sklearn.feature_extraction.text import TfidfVectorizer
text_data_list = customer['Review'].astype(str).tolist( )
text_data_arr = np.array([' '.join(text) for text in text_data.list])
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
text_data_list = customer['Review'].astype(str).tolist()
text_data_arr = np.array([' '.join(text) for text in text_data_list])
```

- tf-idf 값 구하기
  - min_df = 3(단어의 최소 등장 빈도 3번)
  - tokenizer = tokenize (사용자 정의 함수)
  - norm = l2 (엘 2) 을 사용하여 normalization 진행
- fit_transform 함수를 이용하여 배열에 저장된 데이터의 문서단어행렬을 구하고 행렬(matrix) 형식의 변수 text_data에 저장한다

```
vectorizer = TfidfVectorizer(min_df = 3, tokenizer = tokenize, norm = 'l2')
text_data = vectorizer.fit_transform(text_data_arr)
```


```
vectorizer = TfidfVectorizer(min_df=3, tokenizer=tokenize, norm='l2')
text_data = vectorizer.fit_transform(text_data_arr)
```

```
df_tfidf = pd.DataFrame(text_data.A, columns=vectorizer.get_feature_names())
df_tfidf.head()
```

	각	간	감	갑	강	갖	같	개	객	갱	...	효	후	훨	흐	흘
0	0.0	0.0	0.0	0.0	0.083218	0.0	0.107024	0.0	0.0	0.0	...	0.0	0.00000	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.190948	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.00000	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.24791	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.00000	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.000000	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.00000	0.0	0.0	0.0

5 rows × 640 columns

▪ 데이터 시각화

- 데이터 시각화를 위한 라이브러리 import

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

- 한글 폰트 사용을 위한 폰트 설정
  - font_manager, rc import
  - 맑은 고딕 폰트(malgun)를 font_name에 입력
  - rc 함수를 사용해 지정한 폰트 사용

```
from matplotlib import font_manager, rc
font_name = font_manager.FontProperties(fname = “c:/Windows/Fonts/malgun.ttf”)
                                                    .get_name( )
rc(‘font’, family = font_name)
```

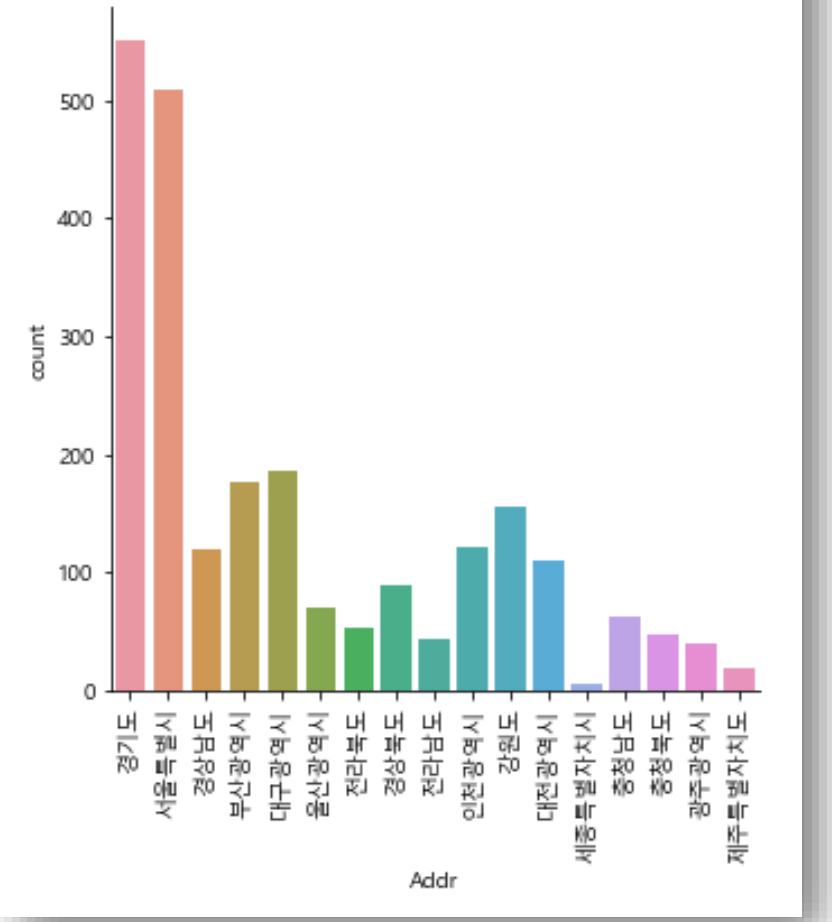
- 주소 데이터
  - Seaborn의 catplot 함수를 이용해 막대 그림 그리기
  - 주소 정보를 기준으로 시각화하므로 Addr 열 사용
  - xticklabels(rotation = 90): x 축의 문자 위치를 90도 회전
  - height(이전 size)

```
graph = sns.catplot(‘Addr’, data = customer, kind = ‘count’, height = 5)
graph.set_xticklabels(rotation = 90)
graph.set_xlabels( )
```

```
from matplotlib import font_manager, rc
font_name = font_manager.FontProperties(fname = "c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family = font_name)
```

```
graph = sns.catplot('Addr', data = customer, kind = 'count', height = 5 )
graph.set_xticklabels(rotation = 90)
graph.set_xlabels()
```

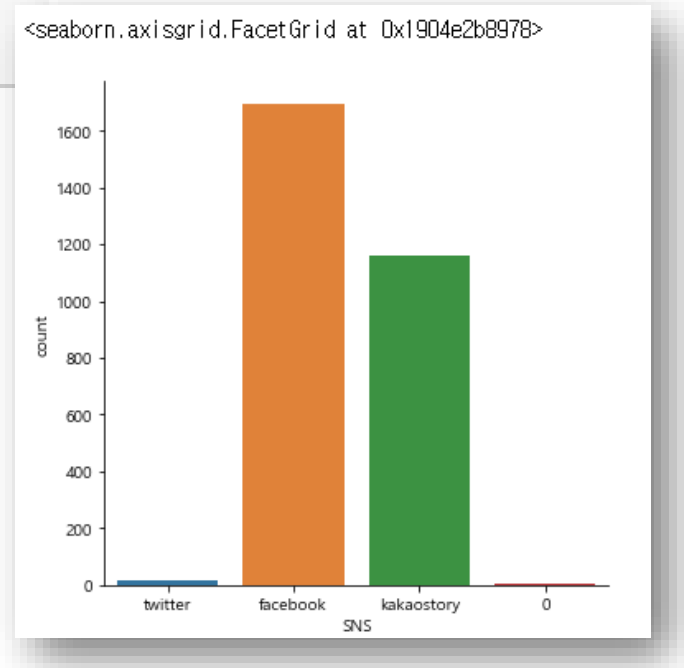
```
<seaborn.axisgrid.FacetGrid at 0x1904e27ad68>
```



- SNS 데이터

```
graph = sns.catplot('SNS', data = customer, kind = 'count', height = 5)
graph.set_xlabels( )
```

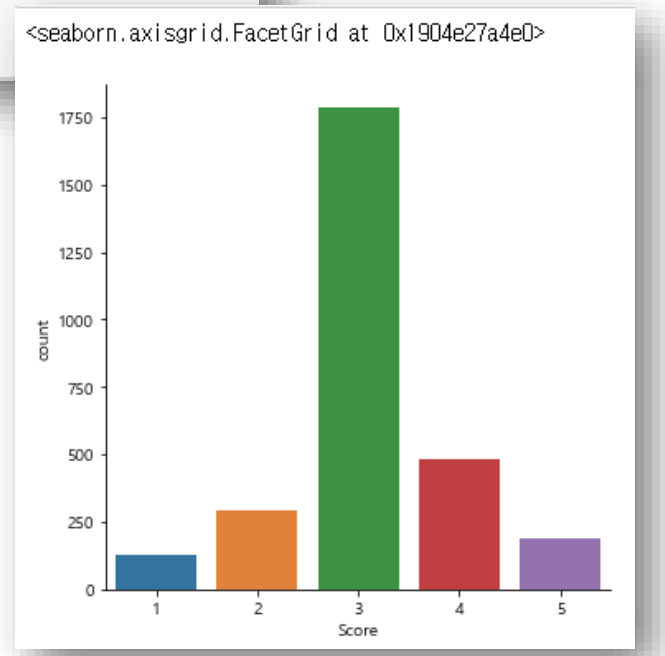
```
graph = sns.catplot('SNS', data = customer, kind = 'count', height = 5)
graph.set_xlabels()
```



- Score 데이터

```
graph = sns.catplot('Score', data = customer, kind = 'count', height = 5)
graph.set_xlabels( )
```

```
graph = sns.catplot('Score', data = customer, kind = 'count', height = 5)
graph.set_xlabels()
```



◦ 3점만 다른 점수에 비해 차이가 많아 조정 필요

- Score 데이터 조정
  - 새로운 score를 위해 Score2 변수 생성 (1,2 점: bad, 3점: normal, 4-5점: good)
  - dropna 메서드: 결측값 제거

```
customer = customer.dropna(subset=['Score'])
customer.index = range(0, len(customer))
customer['Score2'] = ' '
```

```
for i in range(0, len(customer)):
    if(customer['Score'[[i] < 3):
        customer['Score2'[i] = 'bad'
    if(customer['Score'[[i] > 3):
        customer['Score2'[i] = 'good'
    if(customer['Score'[[i] == 3):
        customer['Score2'[i] = 'normal'
customer
```

```
customer = customer.dropna(subset=['Score'])
customer.index = range(0, len(customer))
customer['Score2'] = ''
```

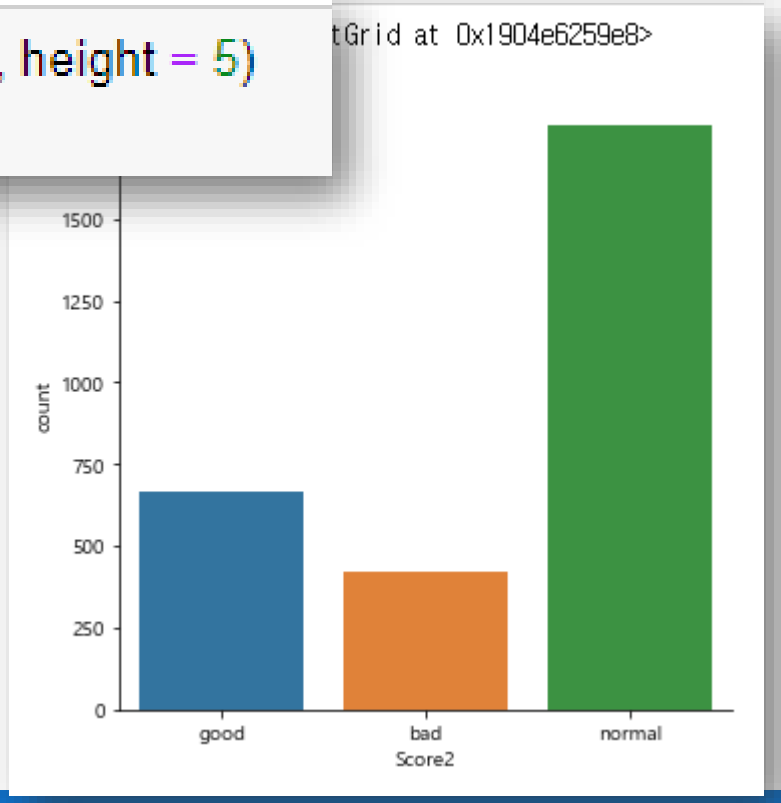
```
for i in range(0, len(customer)) :
    if(customer['Score'][i] < 3) :
        customer['Score2'][i] = 'bad'
    elif(customer['Score'][i] > 3) :
        customer['Score2'][i] = 'good'
    elif(customer['Score'][i] ==3):
        customer['Score2'][i] = 'normal'
customer
```

			Review	SNS	Addr	Score2
			안녕하세요. 데일리 신청해 봅니다. 어렸을 적 부터 장이 좋지 않았는데 고기 인...	twitter	경기도	good
			안그래도 장이 안좋아서 아침마다 고생이거든요~~ 먹고 건강해지고 싶네용 ^^	facebook	서울특별시	bad
			은지 하루종일 더부룩하고 배변후에도 시원하지 않네요.꼭 체험해보고싶습니다.	facebook	서울특별시	bad
3	3	3	우리 아이가 은가를 둥글 둥글 염소 똥처럼 눈답니다. 매번 너무 걱정이랍니다. 꼭 ...	facebook	서울특별시	normal
4	4	5	이것저것 유산균을 먹어봤지만 이거다~ 하는걸 아직 못만났어요. 장이 예민한 우리 작...	kakaostory	경상남도	good
5	5	3	항상 소화가 잘 되지않는 와이프를 위하여 선물하고 싶습니다. 아이 둘 보느라 식사를...	facebook	부산광역시	normal
6	6	1	요즘 넘 피곤해서 체험해 보고 싶어요.	facebook	대구광역시	bad
7	7	1	항상 변비가 있고 소화가 잘 되지 않습니다.	twitter	대구광역시	bad

- Score 수정 후 시각화

```
graph = sns.catplot('Score2', data = customer, kind = 'count', height = 5)
graph.set_xlabels( )
```

```
graph = sns.catplot('Score2', data = customer, kind = 'count', height = 5)
graph.set_xlabels()
```



- 수정된 데이터 세트 저장

```
customer.to_csv(“E:/workspace/customer_data(filtered)_generated.csv’,
                index = False)
```

```
customer.to_csv('E:/workspace/customer_data(filtered)_generated.csv', index=False)
```

3. 데이터 분류

▪ 고객 SNS의 분류 모델 작성

- 사용 데이터 세트 불러오기
  - 수정한 customer_data(filtered)_generated.csv 사용

```
df = pd.read_csv("E:/workspace/customer_data(filtered)_generated.csv")
df.head( )
```

```
df = pd.read_csv("E:/workspace/customer_data(filtered)_generated.csv")
df.head()
```

	Unnamed: 0	Score	Review	SNS	Addr	Score2
0	0	5	안녕하세요. 데일리 신청해 봅니다. 어렸을 적 부터 장이 좋지 않았는데 고기 인...	twitter	경기도	good
1	1	2	오!! 안그래도 장이 안좋아서 아침마다 고생이거든요~~ 먹고 건강해지고 싶네용 ^^	facebook	서울특별시	bad
2	2	2	요즘 장이 안좋은지 하루종일 더부룩하고 배변후에도 시원하지 않네요.꼭 체험해보고싶습니다.	facebook	서울특별시	bad
3	3	3	우리 아이가 은가를 동글 동글 염소 똥처럼 눈답니다. 매번 너무 걱정이랍니다. 꼭 ...	facebook	서울특별시	normal
4	4	5	이것저것 유산균을 먹어봤지만 이거다~ 하는걸 아직 못만났어요. 장이 예민한 우리 작...	kakaostory	경상남도	good

- 데이터를 리스트 형식으로 변형

```
text_data = df[‘Review].astype(str).tolist( )
text_label = df[‘Score’].astype(str).tolist( )
```

```
text_data = df['Review'].astype(str).tolist()
text_label = df['Score2'].astype(str).tolist()
```

- 학습용 데이터 세트와 평가용 데이터 세트로 구분

```
trainset_size = int(round(len(text_data)*0.75))
X_train = np.array([‘ ‘.join(data) for data in text_data[0:trainset_size]])
y_train = np.array([data for data in text_label[0:trainset_size]])

X_test = np.array([‘ ‘.join(data) for data in text_data[trainset_size+1 : len(text_data)]])
y_test = np.array([data for data in text_label[trainset_size+1 : len(text_data)]])
```

  - 학습용과 평가용을 75:25로 구분
  - 알고리즘 적용을 위해 np.array( ) 함수 사용

```
trainset_size = int(round(len(text_data)*0.75))
X_train = np.array([' '.join(data) for data in text_data[0:trainset_size]])
y_train = np.array([data for data in text_label[0:trainset_size]])

X_test = np.array([' '.join(data) for data in text_data[trainset_size+1 : len(text_data)]])
y_test = np.array([data for data in text_label[trainset_size+1 : len(text_data)]])
```

- 문서 단어 행렬 생성(모델 평가 준비)
  - fit_transform 메서드를 이용하여 학습용 데이터 세트 기반으로 문서 단어 행렬 구성
  - transform 메서드를 이용하여 평가용 데이터 세트 기반으로 문서 단어 행렬 구성

```
X_train = vecorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

- 성능 평가 준비

```
df_per = pd.DataFrame(columns = [‘Classifier’, ‘F-Measure’, ‘Accuracy’])
df_per
```

- 성능 평가 패키지 import

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
```


```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
```

▪ Naïve Bayes 알고리즘 적용

- 순서
  - MultinomialNB 패키지 import
  - fit 메서드를 사용하여 모델 학습
  - predict 메서드를 이용해 예측 값 추출

```
from sklearn.naïve_bayes import MultinomialNB
nb_classifier = MultinomialNB( ).fit(X_train, y_train)
nb_pred = nb_classifier.predict(X_test)
```

```
from sklearn.naive_bayes import MultinomialNB
nb_classifier = MultinomialNB().fit(X_train, y_train)
nb_pred = nb_classifier.predict(X_test)
```

```
print("\n Confusion Matrix \n ")
print(confusion_matrix(y_test, nb_pred))
print("\n Classification Report \n")
print(classification_report(y_test, nb_pred))
print(" \n Accuracy \n ")
print(round(accuracy_score(y_test, nb_pred, normalize=True), 2))
```

```
 Confusion Matrix

[[ 11   0  72]
 [  0  10 116]
 [ 12   1 495]]

 Classification Report

              precision    recall  f1-score   support

         bad       0.48      0.13      0.21        83
        good       0.91      0.08      0.15       126
      normal       0.72      0.97      0.83       508

   micro avg       0.72      0.72      0.72       717
   macro avg       0.70      0.40      0.39       717
weighted avg       0.73      0.72      0.64       717


 Accuracy

0.72
```

정리

- KoNLPy 설치
- 네이버 영화 리뷰 분석
- 제약회사 데이터 분석
  - 나이브 베이즈 알고리즘 적용