Aichemist Train Session

CHAP08 텍스트 분석(1)

텍스트 분석(1) 실습 목차

📕Guide-Line

- Github에서 dataset 다운로드 & Kaggle 사이트 참고하기
- 조원과 PPT 내 질문에 답하기
- 배운 텍스트 분석 기법 내용을 떠올려 각 모듈과 기능을 숙지
- Sentiment Analysis 피드백 (개선점, 적용할 수 있는 또다른 기법)

< Women’s E-Commerce Clothing Reviews > Dataset 자료 다운

NICAPOTATO · UPDATED 6 YEARS AGO

1024 | New Notebook | Download (3 MB)

Women's E-Commerce Clothing Reviews

23,000 Customer Reviews and Ratings

Data Card | Code (260) | Discussion (8)

About Dataset

Context

Welcome. This is a Women’s Clothing E-Commerce dataset revolving around the reviews written by customers. Its nine supportive features offer a great environment to parse out the text through its multiple dimensions. Because this is real commercial data, it has been anonymized, and references to the company in the review text and body have been replaced with “retailer”.

Content

This dataset includes 23486 rows and 10 feature variables. Each row corresponds to a customer review, and includes the variables:

- **Clothing ID:** Integer Categorical variable that refers to the specific piece being reviewed.
- **Age:** Positive Integer variable of the reviewers age.
- **Title:** String variable for the title of the review.
- **Review Text:** String variable for the review body.
- **Rating:** Positive Ordinal Integer variable for the product score granted by the customer from 1 Worst, to 5 Best.

Usability
8.82

License
CC0: Public Domain

Expected update frequency
Not specified

Tags

Business | Earth and Nature | Internet | NLP | Ratings and Reviews | Text Mining

https://www.kaggle.com/datasets/nicapotato/womens-ecommerce-clothing-reviews/data

< Women's E-Commerce Clothing Reviews > Dataset 소개

In [2]:
```
df = pd.read_csv("../input/womens-ecommerce-clothing-reviews/Womens Clothing E-Commerce Reviews.csv", index_col=0)
print(df.shape)
df.head(3)
```

```
(23486, 10)
```

Out[2]:

	Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name
0	767	33	NaN	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates
1	1080	34	NaN	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses
2	1077	60	Some major design flaws	I had such high hopes for this dress and reall...	3	0	0	General	Dresses	Dresses

In [3]:
```
df.groupby(['Rating', 'Recommended IND'])['Recommended IND'].count()
```

Out[3]:
```
Rating  Recommended IND
1       0                 826
        1                  16
2       0                1471
        1                  94
3       0                1682
        1                1189
4       0                 168
        1                4909
5       0                  25
        1               13106
Name: Recommended IND, dtype: int64
```

- Review Text : 리뷰 내용이 있는 문자열 변수
- Rating : 1-5 (5에 가까울 수록 좋은 점수)
- Recommend IND : 고객이 제품을 권장하는지를 나타내는 이진 변수, 1은 추천함 0은 추천하지 않음을 의미

02.

Women's E-Commerce Clothing Reviews 워크북

데이터세트 import 및 정보 확인

In [2]:
```
df = pd.read_csv("../input/womens-ecommerce-clothing-reviews/Womens Clothing E-Commerce Reviews.csv", index_col=0)
print(df.shape)
df.head(3)
```

```
(23486, 10)
```

Out[2]:

	Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name
0	767	33	NaN	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates
1	1080	34	NaN	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses
2	1077	60	Some major design flaws	I had such high hopes for this dress and reall...	3	0	0	General	Dresses	Dresses

In [3]:
```
df.groupby(['Rating', 'Recommended IND'])['Recommended IND'].count()
```

Out[3]:
```
Rating  Recommended IND
1       0                826
        1                 16
2       0               1471
        1                 94
3       0               1682
        1               1189
4       0                168
        1               4909
5       0                 25
        1              13106
Name: Recommended IND, dtype: int64
```

데이터세트 전처리 및 타겟값 확인

- Text merging

Merging text features:

```
In [6]: text_df['Review'] = text_df['Title'] + ' ' + text_df['Review Text']
        text_df = text_df.drop(labels=['Title','Review Text'] , axis=1)
        text_df.head()
```

Out[6]:

	Recommended IND	Review
0	1	NaN
1	1	NaN
2	0	Some major design flaws I had such high hopes ...
3	1	My favorite buy! I love, love, love this jumps...
4	1	Flattering shirt This shirt is very flattering...

Q1. 현재 타겟값의 문제점은?

타겟 간의 분포 차이가 큼

- Target Value - positive\negative review

```
In [8]: text_df = text_df[~text_df.Review.isna()]
        text_df = text_df.rename(columns={"Recommended IND": "Recommended"})
        print("My data's shape is:", text_df.shape)
        text_df.head()
```

```
My data's shape is: (19675, 2)
```

Out[8]:

	Recommended	Review
2	0	Some major design flaws I had such high hopes ...
3	1	My favorite buy! I love, love, love this jumps...
4	1	Flattering shirt This shirt is very flattering...
5	0	Not for the very petite I love tracy reese dre...
6	1	Cagrcoal shimmer fun I aded this in my basket ...

Target Value - positive\negative review

```
In [9]: text_df['Recommended'].unique()
```

```
Out[9]: array([0, 1])
```

```
In [10]: text_df['Recommended'].value_counts(normalize=True)
```

```
Out[10]: 1    0.818297
         0    0.181703
         Name: Recommended, dtype: float64
```

데이터세트 피처 추가

1. Text Length

```
In [11]: text_df['Review_length'] = text_df['Review'].apply(len)
         print(text_df.shape)
         text_df.head()
```

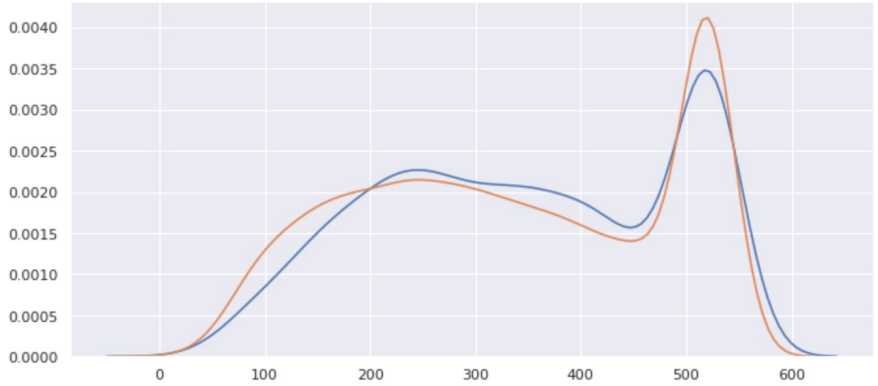
```
(19675, 3)
```

Out[11]:

	Recommended	Review	Review_length
2	0	Some major design flaws I had such high hopes ...	524
3	1	My favorite buy! I love, love, love this jumps...	141
4	1	Flattering shirt This shirt is very flattering...	209
5	0	Not for the very petite I love tracy reese dre...	512
6	1	Cagrcoal shimmer fun I aded this in my basket ...	517

```
In [15]: sns.distplot(df_zero[['Review_length']] ,hist=False)
         sns.distplot(df_one[['Review_length']], hist=False)
```

```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f401ed40a90>
```



리뷰 길이 분포는 그렇게 큰 차이가 있지 않아보인다.

2. Exclamation mark counter

```
In [16]: def count_exclamation_mark(string_text):
             count = 0
             for char in string_text:
                 if char == '!':
                     count += 1
             return count
```

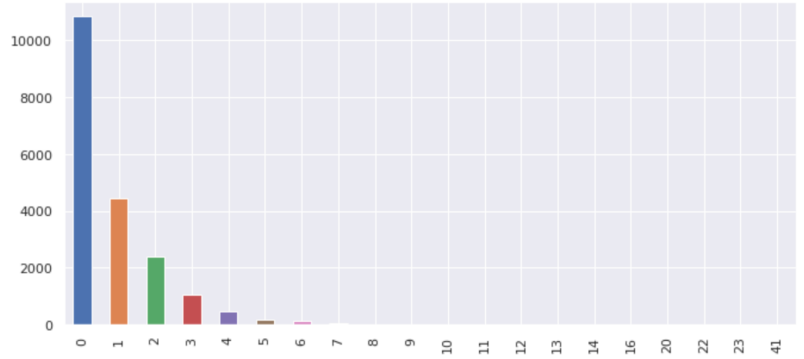
```
In [17]: text_df['count_exc'] = text_df['Review'].apply(count_exclamation_mark)
         text_df.head(5)
```

Out[17]:

	Recommended	Review	Review_length	count_exc
2	0	Some major design flaws I had such high hopes ...	524	1
3	1	My favorite buy! I love, love, love this jumps...	141	3
4	1	Flattering shirt This shirt is very flattering...	209	3
5	0	Not for the very petite I love tracy reese dre...	512	0
6	1	Cagrcoal shimmer fun I aded this in my basket ...	517	0

```
In [19]: text_df['count_exc'].value_counts().sort_index().plot(kind='bar')
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7f401458deb8>
```



Q2. 추천하는 리뷰와 추천하지 않는 리뷰의 Text length 분포를 비교하라

Q3. Exclamation mark을 통해 평가할 수 있는 것은?

긍정, 부정 양쪽 다 느낌표가 가능할 것 같은데, 양극단, 그리고 긍정적인 후기에 조금 더 많을 것 같다

데이터세트 피처 추가

3. Text Polarity

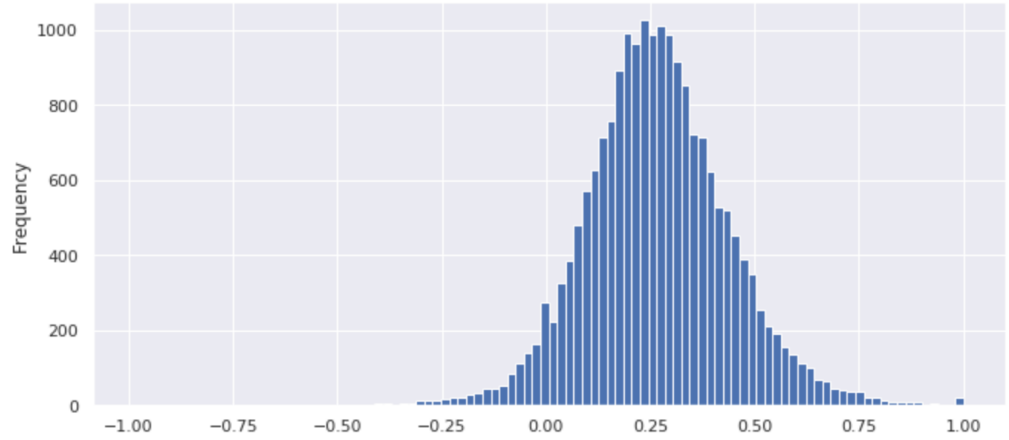
```
In [22]: text_df['Polarity'] = text_df['Review'].apply(lambda x: TextBlob(x).sentiment.polarity)
         text_df.head(5)
```

Out[22]:

	Recommended	Review	Review_length	count_exc	Polarity
2	0	Some major design flaws I had such high hopes ...	524	1	0.073209
3	1	My favorite buy! I love, love, love this jumps...	141	3	0.560714
4	1	Flattering shirt This shirt is very flattering...	209	3	0.512891
5	0	Not for the very petite I love tracy reese dre...	512	0	0.181111
6	1	Cagrcoal shimmer fun I aded this in my basket ...	517	0	0.157500

```
In [23]: text_df['Polarity'].plot(kind='hist', bins=100)
```

```
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7f40141ec630>
```



Q4. Text Polarity feature가 의미하는 게 무엇인가? (TextBlob의 기능을 참고해 답하라)

글에서 나타나는 감정 (음수는 부정의 감정, 양수는 긍정의 감정)

Q5. 양의 값에 치우쳐 있는 건 무엇을 뜻하는가

타겟값 분포가 긍정쪽이 많았어서, text polarity 가 0 이상에 분포가 더 많은 것으로 보인다

데이터세트 전처리 - Stopwords 제거

- Stopwords

In [31]:
```
print(stopwords.words('english')[::12])
```

```
['i', "you'd", 'hers', 'which', 'were', 'a', 'at', 'above', 'again', 'both', 'own', "don't", 'aren', 'haven', "sha
n't"]
```

In [34]:
```
clothes_list =['dress', 'top','sweater','shirt',
               'skirt','material', 'white', 'black',
              'jeans', 'fabric', 'color','order', 'wear']
```

In [35]:
```
def stopwords_removal(messy_str):
    messy_str = word_tokenize(messy_str)
    return [word.lower() for word in messy_str
            if word.lower() not in stop_words and word.lower() not in clothes_list ]
```

In [36]:
```
text_prep['Review'] = text_prep['Review'].apply(stopwords_removal)
text_prep['Review'].head()
```

Out[36]:
```
2    [major, high, wanted, work, ordered, small, us...
3         [favorite, love, love, fabulous, get, great]
4    [flattering, flattering, due, adjustable, perf...
5    [petite, love, reese, petite, tall, long, full...
6    [aded, last, see, look, went, pale, gorgeous, ...
Name: Review, dtype: object
```

Q6. Stopwords이 뭘 의미하고 제거하는 이유는?

판단하기에 의미가 없는 단어
괜히 모델이 헷갈려하지 않도록 없앤다.

데이터세트 전처리 - Stemming

- Stemming

In [43]:
```
porter = PorterStemmer()
```

In [44]:
```
text_prep['Review'] = text_prep['Review'].apply(lambda x: x.split())
text_prep['Review'].head()
```

Out[44]:
```
2    [major, high, wanted, work, ordered, small, us...
3         [favorite, love, love, fabulous, get, great]
4    [flattering, flattering, due, adjustable, perf...
5    [petite, love, reese, petite, tall, long, full...
6    [aded, last, see, look, went, pale, gorgeous, ...
Name: Review, dtype: object
```

In [45]:
```
def stem_update(text_list):
    text_list_new = []
    for word in text_list:
        word = porter.stem(word)
        text_list_new.append(word)
    return text_list_new
```

In [47]:
```
text_prep['Review'] = text_prep['Review'].apply(lambda x: ' '.join(x))
text_prep['Review'].head()
```

Out[47]:
```
2    major high want work order small usual found s...
3                    favorit love love fabul get great
4     flatter flatter due adjust perfect pair cardigan
5    petit love rees petit tall long full overwhelm...
6    ade last see look went pale gorgeou turn mathc...
Name: Review, dtype: object
```

Q7. Stemming 과정으로 text들이 어떻게 변했고

이를 통해 PorterStemmer의 역할을 기술해라

텍스트를 원형으로 변경
단어의 핵심을 파악하는 동시에, 필요 없는 부분 한번 더 제거

데이터세트 시각화 - wordcloud

In [49]:
```
pos_df = text_prep[text_prep.Recommended== 1]
neg_df = text_prep[text_prep.Recommended== 0]
pos_df.head(3)
```

Out[49]:

	Recommended	Review	Review_length	count_exc	Polarity
3	1	favorit love love fabul get great	141	3	0.560714
4	1	flatter flatter due adjust perfect pair cardigan	209	3	0.512891
6	1	ade last see look went pale gorgeou turn mathc...	517	0	0.157500

Positive reviews

```
wordcloud = WordCloud().generate(pos_words)

wordcloud = WordCloud(background_color="white",max_words=len(pos_words),\
                      max_font_size=40, relative_scaling=.5, colormap='summer').generate(pos_words)
plt.figure(figsize=(13,13))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

Bag of Words(BoW)

Bag of Words(BoW)

Bag of Words란 단어들의 순서는 전혀 고려하지 않고, 단어들의 출현 빈도(frequency)에만 집중하는 텍스트 데이터의 수치화 표현 방법

(1) 각 단어에 고유한 정수 인덱스를 부여 # 단어 집합 생성.

(2) 각 인덱스의 위치에 단어 토큰의 등장 횟수를 기록한 벡터를 생성

(예)

```
doc1 = "정부가 발표하는 물가상승률과 소비자가 느끼는 물가상승률은 다르다."
vocab, bow = build_bag_of_words(doc1)
print('vocabulary :', vocab)
print('bag of words vector :', bow)
```

```
vocabulary : {'정부': 0, '가': 1, '발표': 2, '하는': 3, '물가상승률': 4, '과': 5, '소비자': 6, '느끼는': 7,
'은': 8, '다르다': 9}
bag of words vector : [1, 2, 1, 1, 2, 1, 1, 1, 1, 1]
```

https://wikidocs.net/22530

Bag of Words(BoW)

```
In [54]: def text_vectorizing_process(sentence_string):
             return [word for word in sentence_string.split()]

In [55]: bow_transformer = CountVectorizer(text_vectorizing_process)

In [56]: bow_transformer.fit(text_prep['Review'])
```

```
In [59]: Reviews = bow_transformer.transform(text_prep['Review'])
         Reviews

Out[59]: <19675x7277 sparse matrix of type '<class 'numpy.int64'>'
                 with 237960 stored elements in Compressed Sparse Row format>

In [60]: print('Shape of Sparse Matrix', Reviews.shape)
         print('Amount of Non-Zero occurences:', Reviews.nnz)

         Shape of Sparse Matrix (19675, 7277)
         Amount of Non-Zero occurences: 237960
```

DTM (문서 단어 행렬)

```
In [57]: print(text_prep['Review'].iloc[3])

         petit love rees petit tall long full overwhelm small shorten narrow take love work return

In [58]: example = bow_transformer.transform([text_prep['Review'].iloc[3]])
         print(example)

           (0, 2416)     1
           (0, 3438)     1
           (0, 3507)     2
           (0, 3881)     1
           (0, 4302)     1
           (0, 4438)     2
           (0, 4875)     1
           (0, 5004)     1
           (0, 5383)     1
           (0, 5601)     1
           (0, 6196)     1
           (0, 6201)     1
           (0, 7159)     1
```

위 57번에서 각 단어의 언급 횟수가 수치화되어있다

Q8. 58번 행의 결과가 왼쪽과 같이 나오는 이유를 설명해라

문서 벡터화 : TF-IDF

TF-IDF (Term Frequency–Inverse Document Frequency**)**

TF-IDF(Term Frequency-Inverse Document Frequency)는 단어의 빈도와 역 문서 빈도(문서의 빈도에 특정 식을 취함)를 사용하여 DTM 내의 각 단어들마다 중요한 정도를 가중치로 주는 방법. TF-IDF는 TF와 IDF를 곱한 값을 의미 //여기까지만 알기

```
In [61]: tfidf_transformer = TfidfTransformer().fit(Reviews)

         tfidf_example = tfidf_transformer.transform(example)
         print (tfidf_example)
```

```
In [65]: print(messages_tfidf[:1])
```

```
  (0, 7273)     0.21472646647428087
  (0, 7159)     0.12712196996857394
  (0, 6930)     0.12017260570324308
  (0, 6813)     0.1513403250714934
  (0, 6383)     0.14060642619885752
  (0, 5601)     0.29785858576976637
  (0, 5290)     0.18845614450376966
  (0, 4943)     0.2444427546112772
  (0, 4438)     0.13986672876492434
  (0, 4260)     0.15858178150697497
  (0, 4210)     0.1001757163619286
  (0, 4138)     0.20901503445074096
  (0, 3944)     0.30397328049150685
  (0, 3584)     0.5250571838807988
  (0, 2934)     0.3111586573431781
  (0, 2769)     0.14606384138147635
  (0, 2364)     0.15056928107008546
  (0, 2197)     0.08515558755200825
  (0, 1153)     0.10625437706817653
  (0, 984)      0.19198542692781725
  (0, 698)      0.18408106529286453
```

Q9. TF-IDF 과정은 왜 필요한가?

너무 흔한 단어에 모델이 비중을 크게 두며 학습하는 것을 방지하기 위해 과도한 다빈도수는 조금의 감소를 시켜두며 예측의 성능을 높인다.

데이터세트 타겟값 처리

Q10. Target 데이터은 어떤 feature에 해당되는가?

recommend
(x에서는 삭제시키고, y(정답지)에는 남겨둠)

In [68]:
```
X = df_all.drop('Recommended', axis=1)
y = df_all.Recommended

X.head()
```

Out[68]:

	Review_length	count_exc	Polarity	0	1	2	3	4	5	6	...	7267	7268	7269	7270	7271	7272	7273	7274	7275	7276
2	524	1	0.073209	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	141	3	0.560714	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	209	3	0.512891	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	512	0	0.181111	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	517	0	0.157500	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Text

In [72]:
```
y_train.value_counts(normalize=True)
```

Out[72]:
```
1    0.818741
0    0.181259
Name: Recommended, dtype: float64
```

In [73]:
```
y_test.value_counts(normalize=True)
```

Out[73]:
```
1    0.818605
0    0.181395
Name: Recommended, dtype: float64
```

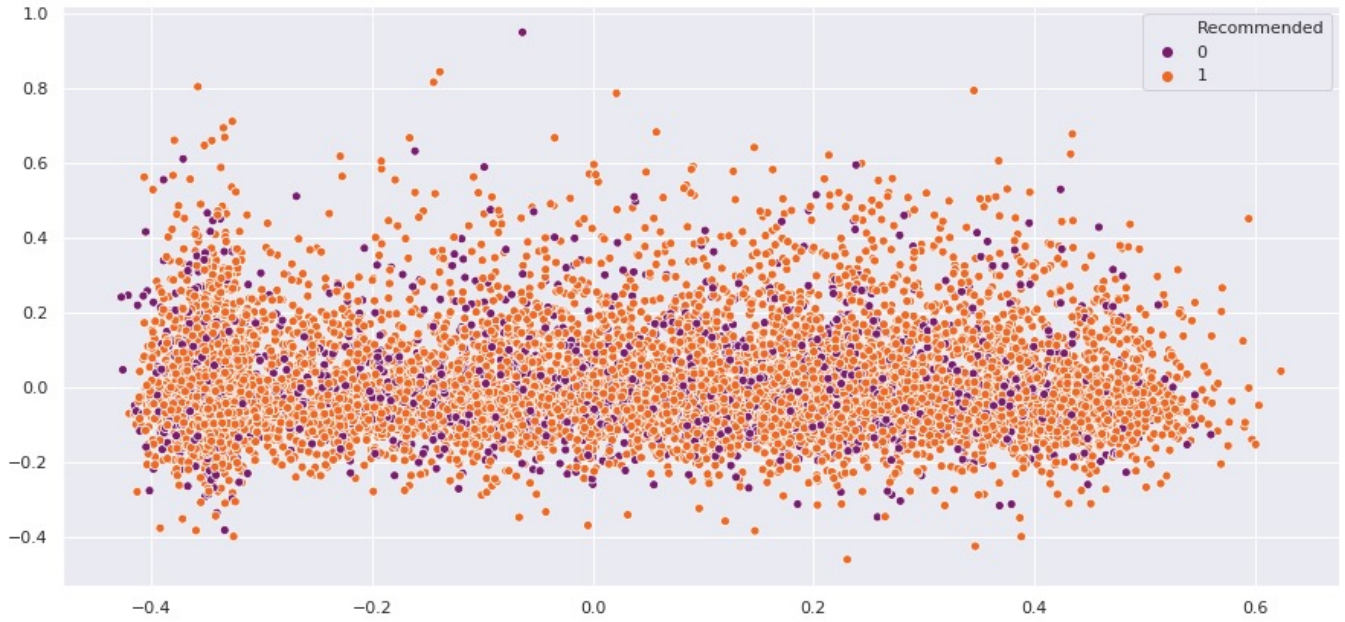
Q11. y_target, y_train 데이터의 분포 문제점은?

어떻게 해결할 수 있는가? (notebook 참고)

언더샘플링, 오버샘플링, 교차검증

데이터세트 차원 축소

```
In [76]: pca_transformer = PCA(n_components=2).fit(X_train_scaled)
         X_train_scaled_pca = pca_transformer.transform(X_train_scaled)
         X_test_scaled_pca = pca_transformer.transform(X_test_scaled)
         X_train_scaled_pca[:1]

Out[76]: array([[-0.13600041, -0.04967514]])
```



칼럼이 커졌기 때문에 차원축소 진행
(7천개 -> 2개로)

X_train_scaled_pca[:0]이 1번째 축으로 그래프의 x축

X_train_scaled_pca[:1]이 2번째 축으로 그래프의 y축

>> 2개의 PCA 컴포넌트 만으로 여러 feature 변동성을 설명할 수 있음.

Data를 2개의 축으로 visualization하기 위해 X data를 2개의 축으로 차원축소

>> PCA를 통해 data visualization

모델 학습/예측/평가

1. SVM

In [80]:
```
svc_model = SVC(C=1.0,
                kernel='linear',
                class_weight='balanced',
                probability=True,
                random_state=111)
svc_model.fit(X_train_scaled, y_train)
```

Out[80]:
```
SVC(C=1.0, cache_size=200, class_weight='balanced', coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=True, random_state=111,
    shrinking=True, tol=0.001, verbose=False)
```

In [81]:
```
test_predictions = svc_model.predict(X_test_scaled)
print(report(y_test, test_predictions, svc_model.classes_ ))
```

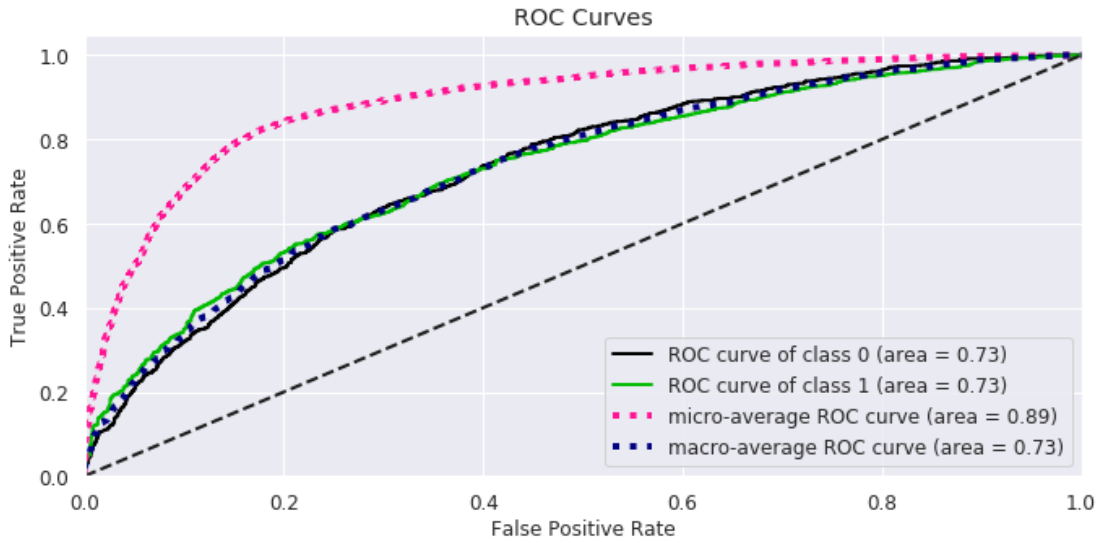
```
Confusion Matrix:
     0     1
0  517   380
1  995  3053

Classification Report:
              precision    recall  f1-score   support

           0       0.34      0.58      0.43       897
           1       0.89      0.75      0.82      4048

    accuracy                           0.72      4945
   macro avg       0.62      0.67      0.62      4945
weighted avg       0.79      0.72      0.75      4945
```

Q12. 각 모델들은 무엇을 예측하는 건가요? (Sentiment Analysis의 최종 목적)

긍정/부정 감정을 리뷰에서 예측하는 모델

Q13. 이때 TF-IDF 가중치 결과가 어떻게 사용되었는가?

불필요한 빈도수 높은 단어를 모델이 무시할 수 있도록



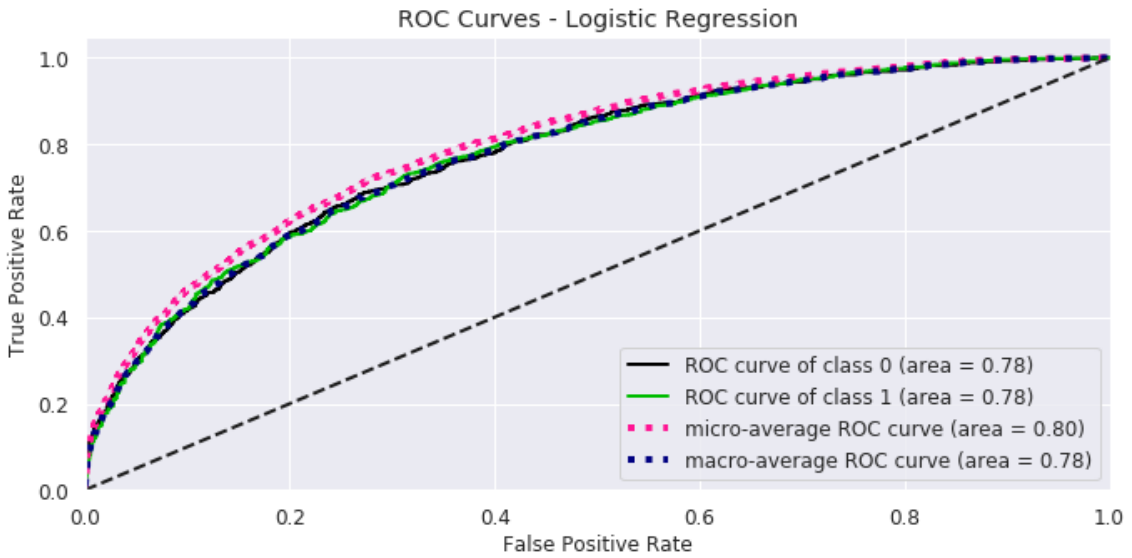
모델 학습/예측/평가

2. Logistic Regression

In [83]:
```
lr_model = LogisticRegression(class_weight='balanced',
                              random_state=111,
                              solver='lbfgs',
                              C=1.0)

gs_lr_model = GridSearchCV(lr_model,
                           param_grid={'C': [0.01, 0.1, 0.5, 1.0, 5.0]},
                           cv=5,
                           scoring='roc_auc')

gs_lr_model.fit(X_train_scaled, y_train)
```

In [85]:
```
test_predictions = gs_lr_model.predict(X_test_scaled)
print(report(y_test, test_predictions, gs_lr_model.classes_ ))
```

```
Confusion Matrix:
      0     1
0   617   280
1  1100  2948

Classification Report:
              precision    recall  f1-score   support

           0       0.36      0.69      0.47       897
           1       0.91      0.73      0.81      4048

    accuracy                           0.72      4945
   macro avg       0.64      0.71      0.64      4945
weighted avg       0.81      0.72      0.75      4945
```



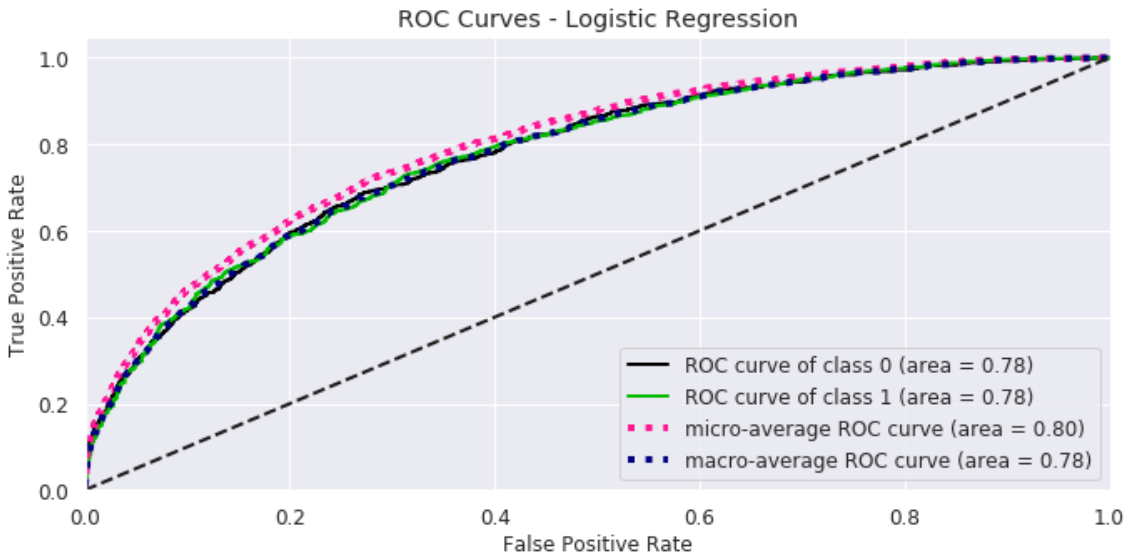
모델 학습/예측/평가

2. Logistic Regression

In [83]:
```
lr_model = LogisticRegression(class_weight='balanced',
                              random_state=111,
                              solver='lbfgs',
                              C=1.0)

gs_lr_model = GridSearchCV(lr_model,
                           param_grid={'C': [0.01, 0.1, 0.5, 1.0, 5.0]},
                           cv=5,
                           scoring='roc_auc')

gs_lr_model.fit(X_train_scaled, y_train)
```

In [85]:
```
test_predictions = gs_lr_model.predict(X_test_scaled)
print(report(y_test, test_predictions, gs_lr_model.classes_ ))
```

```
Confusion Matrix:
      0     1
0   617   280
1  1100  2948

Classification Report:
              precision    recall  f1-score   support

           0       0.36      0.69      0.47       897
           1       0.91      0.73      0.81      4048

    accuracy                           0.72      4945
   macro avg       0.64      0.71      0.64      4945
weighted avg       0.81      0.72      0.75      4945
```



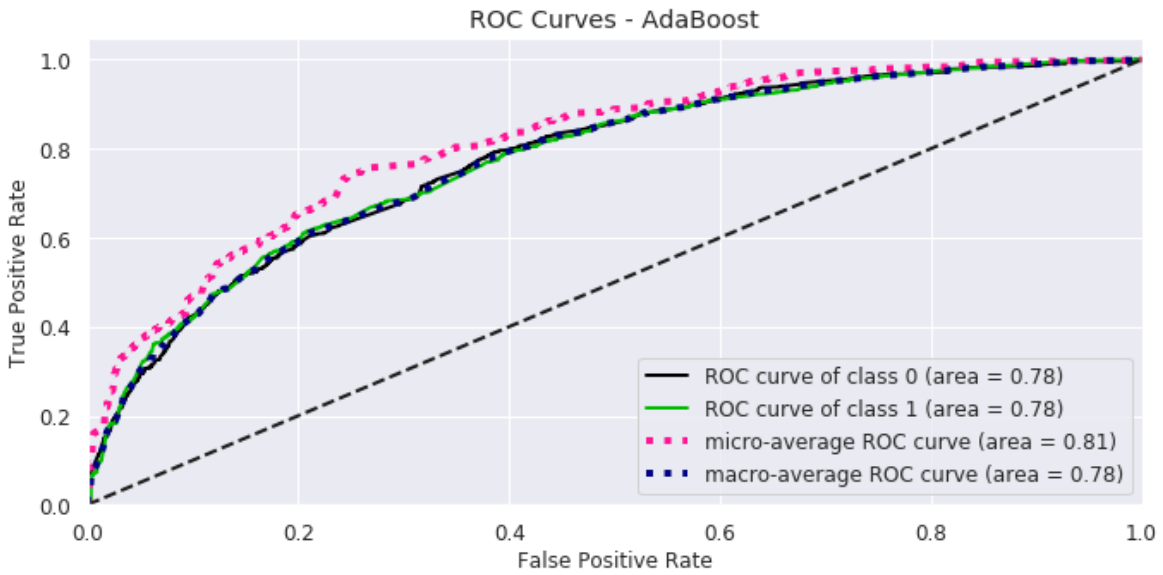
모델 학습/예측/평가

3. AdaBoost

In [87]:
```
dt = DecisionTreeClassifier(max_depth=5, class_weight='balanced', random_state=555)

ada_model = AdaBoostClassifier(base_estimator=dt, learning_rate=0.001, n_estimators=1000, random_state=222)
ada_model.fit(X_train ,y_train)
```

In [88]:
```
test_predictions = ada_model.predict(X_test)
print(report(y_test, test_predictions, ada_model.classes_ ))
```

```
Confusion Matrix:
     0     1
0  548   349
1  909  3139

Classification Report:
              precision    recall  f1-score   support

           0       0.38      0.61      0.47       897
           1       0.90      0.78      0.83      4048

    accuracy                           0.75      4945
   macro avg       0.64      0.69      0.65      4945
weighted avg       0.80      0.75      0.77      4945
```



모델 학습/예측/평가

4. Random Forest

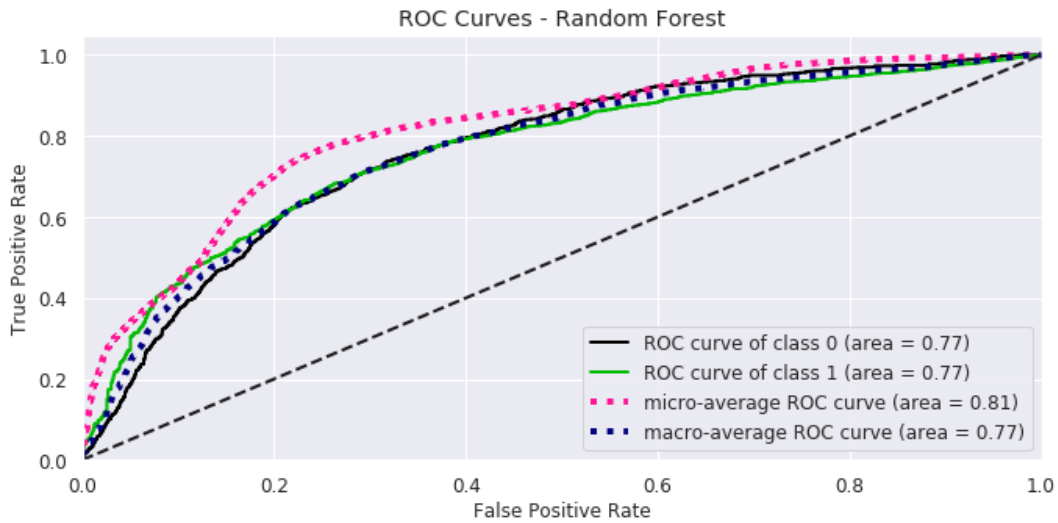
```
In [90]: rf_model = RandomForestClassifier(n_estimators=1000, max_depth=5,
                                           class_weight='balanced', random_state=3)
         rf_model.fit(X_train, y_train)
```

```
Out[90]: RandomForestClassifier(bootstrap=True, class_weight='balanced',
                                criterion='gini', max_depth=5, max_features='auto',
                                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,
                                n_estimators=1000, n_jobs=None, oob_score=False,
                                random_state=3, verbose=0, warm_start=False)
```

```
In [91]: test_predictions = rf_model.predict(X_test)
         print(report(y_test, test_predictions, rf_model.classes_ ))
```

```
Confusion Matrix:
     0     1
0  533   364
1  829  3219

Classification Report:
              precision    recall  f1-score   support

           0       0.39      0.59      0.47       897
           1       0.90      0.80      0.84      4048

    accuracy                           0.76      4945
   macro avg       0.64      0.69      0.66      4945
weighted avg       0.81      0.76      0.78      4945
```

Q14. f1 score로 평가할 때 가장 예측 성능이 좋은 모델은?



수고하셨습니다