네이버 영화 평점 분석 및 긍부정 예측 모델링

- 네이버 영화 평점: https://movie.naver.com/movie/point/af/list.nhn (https://movie.naver.com/movie/point/af/list.nhn)
- 데이터: https://github.com/e9t/nsmc (https://github.com/e9t/nsmc)

데이터셋 읽기

In [84]:
```
def read_file(filename):
    with open(filename, 'rt', encoding='utf-8') as f:
        review_file = [line.split('\t') for line in f.read().splitlines()]
        review_file = review_file[1:]

    return review_file
```

In [85]:
```
review_train = read_file('nsmc-master/ratings_train.txt')
review_test = read_file('nsmc-master/ratings_test.txt')
```

데이터셋 확인

In [60]:
```
print(len(review_train))
review_train[:5]
```

```
150000
```

Out[60]:
```
[['9976970', '아 더빙.. 진짜 짜증나네요 목소리', '0'],
 ['3819312', '흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나', '1'],
 ['10265843', '너무재밋었다그래서보는것을추천한다', '0'],
 ['9045019', '교도소 이야기구먼 ..솔직히 재미는 없다..평점 조정', '0'],
 ['6483659',
  '사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 던스트가 너무나도 이뻐보였다',
  '1']]
```

In [61]:
```
print(len(review_test))
review_test[:5]
```

```
50000
```

Out[61]:
```
[['6270596', '굳 ㅋ', '1'],
 ['9274899', 'GDNTOPCLASSINTHECLUB', '0'],
 ['8544678', '뭐야 이 평점들은.... 나쁘진 않지만 10점 짜리는 더더욱 아니잖아', '0'],
 ['6825595', '지루하지는 않은데 완전 막장임... 돈주고 보기에는....', '0'],
 ['6723715', '3D만 아니었어도 별 다섯 개 줬을텐데.. 왜 3D로 나와서 제 심기를 불편하게 하죠??', '0']]
```

텍스트 전처리

In [74]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import re
import nltk
import string
from nltk import Text, FreqDist
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import konlpy
from konlpy.tag import Okt
```

In [96]:
```
#한글 불용어 리스트 (인터넷 검색 및 추가)
with open('nsmc-master/한국어불용어.txt', 'rt', encoding='utf-8') as f:
    sw = f.read().splitlines()

sw[-15:]
```

Out[96]:
```
['ㅆ', 'ㅉ', 'ㅏ', 'ㅑ', 'ㅓ', 'ㅛ', 'ㅜ', 'ㅠ', 'ㅡ', 'ㅣ', 'ㅐ', 'ㅔ', 'ㅒ', 'ㅖ', '흠']
```

In [102]:
```
def text_preprocessing(document):
    #특수문자 제거
    pattern = '[{}]'.format(string.punctuation)
    document = re.sub(pattern, ' ', document)

    #토큰화, 불용어제거 및 어간추출
    okt = Okt()
    doc_morph = okt.morphs(document, stem=True)
    result_token = [word_token for word_token in doc_morph if word_token not in sw]

    #return result_token
    return ' '.join(result_token)
```

In [112]:
```
# 데이터 분류 --> input: X, output: y
X_train = [line[1] for line in review_train]
y_train = [line[2] for line in review_train]

X_test = [line[1] for line in review_test]
y_test = [line[2] for line in review_test]
```

In [134]:
```
print(len(y_train))
y_train[:5]
```

```
150000
```

Out[134]:
```
['0', '1', '0', '0', '1']
```

In [115]:
```
X_train[:5] #텍스트 전처리 전
```

Out[115]:
```
['아 더빙.. 진짜 짜증나네요 목소리',
 '흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나',
 '너무재밋었다그래서보는것을추천한다',
 '교도소 이야기구먼 ..솔직히 재미는 없다..평점 조정',
 '사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 던스트가 너무나도 이뻐보였다']
```

In [116]:
```
X_train = [text_preprocessing(sent) for sent in X_train]
print(len(X_train))
X_train[:10] #텍스트 전처리 후
```

```
150000
```

Out[116]:
```
['더빙 진짜 짜증나다 목소리',
 '포스터 보고 초딩 영화 줄 오버 연기 가볍다 않다',
 '무재 밋었 다그 래서 보다 추천 한 다',
 '교도소 이야기 구먼 솔직하다 재미 는 없다 평점 조정',
 '사이 몬페 익살스럽다 연기 돋보이다 영화 스파이더맨 늙다 보이다 하다 커스틴 던스트 너무나도 이쁘다 보이다',
 '막 걸음 마 떼다 3 세 초등학교 1 학년 생인 8 살다 영화 ㅋㅋㅋ 별 반개 도 아깝다 움',
 '원작 긴장감 제대로 살리다 하다',
 '별 반개 도 아깝다 욕 나오다 이응경 길용우 연 기 생활 인지 정말 발 해도 그것 보단 낫다 납치 감금 만 반복 반복 드라마 는 가족 도 없다
연기 못 하다 사람 만 엿',
 '액션 없다 재미 안되다 영화',
 '왜케 평점 낮다 꽤 볼 만 한 데 헐리우드 식 화려하다 너무 길들이다']
```

In [135]:
```
print(len(y_test))
y_test[:5]
```

```
50000
```

Out[135]:
```
['1', '0', '0', '0', '0']
```

In [117]:
```
X_test[:5] #텍스트 전처리 전
```

Out[117]:
```
['굳 ㅋ',
 'GDNTOPCLASSINTHECLUB',
 '뭐야 이 평점들은.... 나쁘진 않지만 10점 짜리는 더더욱 아니잖아',
 '지루하지는 않은데 완전 막장임... 돈주고 보기에는....',
 '3D만 아니었어도 별 다섯 개 줬을텐데.. 왜 3D로 나와서 제 심기를 불편하게 하죠??']
```

In [118]:
```
X_test = [text_preprocessing(sent) for sent in X_test]
print(len(X_test))
X_test[:10] #텍스트 전처리 후
```

```
50000
```

Out[118]:
```
['굳다',
 'GDNTOPCLASSINTHECLUB',
 '뭐 평점 은 나쁘다 않다 10 점 짜다 리 는 더 더욱 아니다',
 '지루하다 않다 완전 막장 임 돈 주다 보기 에는',
 '3 D 만 아니다 별 개 주다 3 D 나오다 심기 불편하다 하다',
 '음악 주가 되다 최고 음악 영화',
 '진정하다 쓰레기',
 '미국 애니 튀어나오다 한 창의력 없다 로봇 디자인 부터가 고개 젖다 하다',
 '갈수록 개판 되다 중국영화 유치하다 내용 없다 폼 잡다 끝나다 말 도 안되다 무기 유치하다 cg 남무 그리다 동사서독 영화 이건 3 류 류작 이다',
 '이별 아픔 뒤 찾아오다 새롭다 인연 기쁨 But 모든 사람 그렇다 않다']
```

Feature Vectorization

In [145]:
```
#CountVectorizer

from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(min_df=10, ngram_range=(1,5))
cv.fit(X_train)
X_train_dtm = cv.transform(X_train)
X_test_dtm = cv.transform(X_test)
```

In [146]:
```
X_train_dtm.shape, X_test_dtm.shape
```

Out[146]:
```
((150000, 20915), (50000, 20915))
```

In [152]:
```
#TfidfVectorizer

from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(min_df=10, ngram_range=(1,5))
tfidf.fit(X_train)
X_train_tfidf = tfidf.transform(X_train)
X_test_tfidf = tfidf.transform(X_test)
```

In [153]:
```
X_train_tfidf.shape, X_test_tfidf.shape
```

Out[153]:
```
((150000, 20915), (50000, 20915))
```

긍부정 예측 머신러닝 모델링

CountVectorizer로 생성된 DTM --> LogisticRegression

In [147]:
```
#CountVectorizer --> LogisticRegression

from sklearn.linear_model import LogisticRegression

lg_clf = LogisticRegression(max_iter=1000)
lg_clf.fit(X_train_dtm, y_train)
```

Out[147]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [148]:
```
#분류 예측
pred_train = lg_clf.predict(X_train_dtm)
pred_test = lg_clf.predict(X_test_dtm)

#확률
pred_train_proba = lg_clf.predict_proba(X_train_dtm)
pred_test_proba = lg_clf.predict_proba(X_test_dtm)
```

In [149]:
```
#평가
from sklearn.metrics import accuracy_score, roc_auc_score

print("훈련데이터 예측 평가")
print("정확도: {}".format(accuracy_score(y_train, pred_train)), "AUC: {}".format(roc_auc_score(y_train, pred_train_proba[:,1])))
print("테스트데이터 예측 평가")
print("정확도: {}".format(accuracy_score(y_test, pred_test)), "AUC: {}".format(roc_auc_score(y_test, pred_test_proba[:,1])))
```

```
훈련데이터 예측 평가
정확도: 0.8824266666666667 AUC: 0.9555584657616571
테스트데이터 예측 평가
정확도: 0.83686 AUC: 0.9179532478764092
```

TF-IDF/TfidfVectorizer로 생성된 DTM --> LogisticRegression

In [154]:
```
#TfidfVectorizer --> LogisticRegression
from sklearn.linear_model import LogisticRegression

lg_clf2 = LogisticRegression(max_iter=1000)
lg_clf2.fit(X_train_tfidf, y_train)
```

Out[154]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [155]:
```
pred_train2 = lg_clf2.predict(X_train_tfidf)
pred_test2 = lg_clf2.predict(X_test_tfidf)

pred_train_proba2 = lg_clf2.predict_proba(X_train_tfidf)
pred_test_proba2 = lg_clf2.predict_proba(X_test_tfidf)
```

In [156]:
```
from sklearn.metrics import accuracy_score, roc_auc_score

print("훈련데이터 예측 평가")
print("정확도: {}".format(accuracy_score(y_train, pred_train2)), "AUC: {}".format(roc_auc_score(y_train, pred_train_proba2[:,1])))
print("테스트데이터 예측 평가")
print("정확도: {}".format(accuracy_score(y_test, pred_test2)), "AUC: {}".format(roc_auc_score(y_test, pred_test_proba2[:,1])))
```

```
훈련데이터 예측 평가
정확도: 0.8695 AUC: 0.9429230522211607
테스트데이터 예측 평가
정확도: 0.84116 AUC: 0.921389039604106
```

In [44]:
```
#새 데이터로 예측
new_review =[
    "초반부는 흥미진진한데 가면 갈수록 루즈해짐",
    "정말로 재미있었어요",
    "시간때우려 영화관갔다가 울면서 나왔다"
]
```

In [45]:
```
#텍스트 전처리
new_review = [text_preprocessing(review) for review in new_review]
new_review
```

Out[45]:
```
['초반 불다 흥미진진 한 데 가면 갈수록 루즈 하다 짐', '정말로 재미있다', '시간 때우다 영화관 갓 다가 울면 서 나오다']
```

In [48]:
```
new_X = tfidf.transform(new_review) #TfidfVectorizer
```

In [49]:
```
lg_clf.predict(new_X)
```

Out[49]:
```
array([0, 1, 0], dtype=int64)
```