[STA518 기말프로젝트]

Sentiment Analysis with Machine Learning Models:

Logistic Regression, Support Vector Machine, Decision Tree, and Random Forest

2020-2학기 통계분석방법론
2020021218 통계학과
이소연

erinlee42@korea.ac.kr
2020/12/21

목 차

제 1 장. 프로젝트 개요 3

1.1. 감성분석이란?

1.2. 프로젝트 설계

1.3. 데이터 소개

1.4. 모델 평가 지표

1.4.1 정확도

1.4.2 F1-값

1.4.3 PR-AUC와 ROC-AUC

제 2 장. 데이터 분석 11

2.1. 텍스트 전처리

2.2. VADER

2.3. Bag-of-Words

2.4. TF-IDF

2.5. 모형 구축

2.5.1. 로지스틱회귀

2.5.2. 서포트벡터머신

2.5.3. 의사결정나무

2.5.4. 랜덤포레스트

제 3 장. 결론 35

3.1. 분석 결과 비교

3.2. 향후 연구 방향

부록 42

1. 참고 문헌

2. 코드

제 1 장. 프로젝트 개요

1.1. 감성분석이란?

사람의 '감성'에 대한 정의는 학문 분야에 따라 매우 다양하다. 그렇다면 텍스트 마이닝의 관점에서 '감성'은 어떻게 정의될까?

텍스트 마이닝은 텍스트 형태의 비정형 데이터에 분석 기술을 적용한 것으로, 텍스트에 포함된 단어들을 분석하여 단어 간의 관계를 파악하고, 정보를 찾아내는 기법이다. 그중 감성 분석은 텍스트에 숨겨져 있는 저자의 의도나 정보를 찾아내고자 한다. 즉 '감성'이란, 문서에 내포된 작성자의 주요한 기분, 상태를 뜻한다. 대표적인 예로, 온라인 쇼핑몰에서의 사용자의 상품평 분석이 있다. 이처럼 감성 분석은 주로 기업이 고객의 의견을 수렴하는데 효율적이며, 특히 소비자의 니즈에 부합하는 마케팅 전략이나, 주식 시장의 동향에 따른 계획을 세울 때 유용하다.

감성 분석은 크게 어휘기반 접근방법(Lexicon-based approach)과 기계학습기반 접근방법(Machine Learning-based approach)가 있다. 먼저 어휘기반 접근방법은 "기쁨", "슬픔"과 같이 확실히 알려진 어구, 어미, 혹은 관용 표현 등을 활용하여 데이터를 평가하는 방법이다. 이 중 하나인 사전기반 접근방법(Dictionary-based Approach)은 주어진 텍스트를 분석 및 추출하여 긍정적인 단어와 부정적인 단어를 구분한 '감성 사전'을 구축하고, 이를 기반으로 새로운 문서에 대해 분석을 한다. 하지만 데이터가 기하급수적으로 증가하고, 새로운 개념들이 빠르게 생기며 이러한 접근 방식은 한계에 부딪히게 되었다. 이를 보완한 것이 바로 기계학습기반 접근방법이다. 이는 수집된 텍스트 데이터에 기계학습의 분류 알고리즘을 적용한다. 그 과정은 pre-trained Language Model을 활용한 비지도학습 기법(Unsupervised Learning method)과 지도학습 기법(Supervised Learning method)으로 나뉘는데, 이 중 성능 면에선 지도학습 기법이 월등히 높을 수밖에 없다. 또한, 최근에는 사전기반 접근방법과 기계학습기반 접근방법을 혼합하는 경우가 가장 흔하다. 예컨

대, 사전에 구축된 '감성 사전'을 기반으로 감성점수(Sentiment Score)를 산출하여, 기존 머신러닝 알고리즘으로 텍스트의 패턴을 찾는 것이다. 따라서, 본 프로젝트 또한 지도학습에 기반한 혼합 접근법을 사용할 것이다.

1.2. 프로젝트 설계

본 프로젝트는 [통계분석방법론] 강의에서 배운 통계적 기계학습 모형을 필자의 관심 연구 분야인 텍스트 마이닝에 적용하고자 한다.

따라서, 본 연구에선 주어진 데이터에 대해 감성 분석을 수행하기 위해 기계학습기반 접근방법에서 가장 활발하게 쓰이는 로지스틱 회귀(Logistic regression)과 서포트 벡터 머신(Support Vector Machine), 그리고 수업에서 다룬 또 다른 분류 모델인 의사결정나무(Decision Tree)와 랜덤포레스트(Random Forest) 앙상블 모형을 사용할 것이다. 각 모형의 원리와 감성 분석 적용 과정을 탐구하고, 나아가 분류 및 예측 성능을 비교할 것이다. 모델 성능의 공정한 비교를 위해 총 10번의 시뮬레이션을 반복할 것이다. 모델 구축을 위해 참고한 교재의 단원은 다음과 같다.

[생명과학 연구를 위한 통계적 방법] 4장. 로짓 및 로그선형 분석

[응용데이터분석] 17장. 서포트벡터머신

[응용데이터분석] 18장. 나무모형

이러한 모형을 적용하기 위해선, 텍스트 데이터에서 추출한 단어들을 설명변수로, 해당 텍스트에 대한 감성을 양/음으로 분류한 레이블(0 혹은 1)을 목적변수로 두어야 한다. 즉, 문서의 변수화 작업이 필요하다. 텍스트를 수치형 피처 벡터로 표현하는 방법은 여러 개가 있는데, 가장 기본적인 방법으로 Bag-of-Words 모델과 이보다 좀 더 복잡한 TF-IDF 모델이 있다. 자세한 설명은 다음 장에서 하도록 한다. 다음으로, 목적변수의 경우 각 리뷰의 별점뿐만 아니라, 소셜 미디어 전용 오픈소스 감성분석 툴인 VADER(Valence Aware Dictionary and sEntiment Reasoner)를 활용하여 각 문장을 이항의 '감성(Sentiment)'으로 나타내겠다.

1.3. 데이터 소개

본 프로젝트에선 10년간 축적된 Amazon의 Fine Foods에 관한 리뷰 데이터(http://snap.stanford.edu/data/web-FineFoods.html)를 사용할 것이다. 원자료를 파이썬 데이터 프레임으로 가공하면, 아래와 같은 형태를 띈다.

```
df.head(6)
```

Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	1219017600	"Delight" says it all	This is a confection that has been around a fe...
4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	1350777600	Great taffy	Great taffy at a great price. There was a wid...
6	B006K2ZZ7K	ADT0SRK1MGOEU	Twoapennything	0	0	4	1342051200	Nice Taffy	I got a wild hair for taffy and ordered this f...

ProductId: 제품 고유 식별 코드

UserId: 작성자 아이디

ProfileName: 작성자 이름

HelpfulnessNumerator: 추천 수

HelpfulnessDenominator: 추천 수와 비추천 수를 합한 값

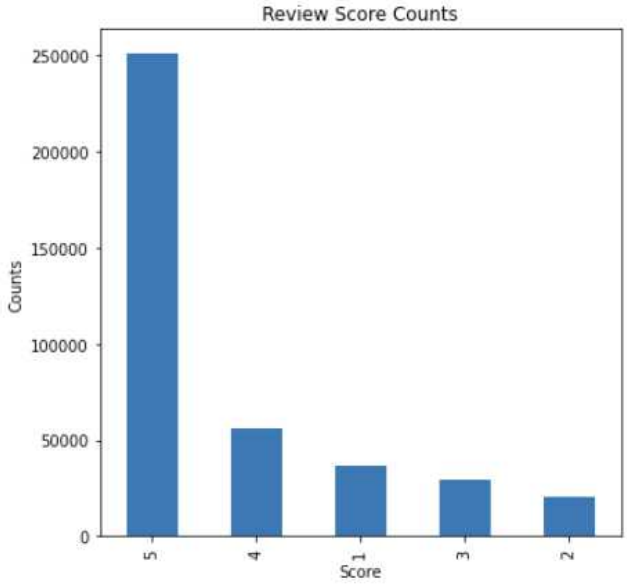
Score: 리뷰 별점

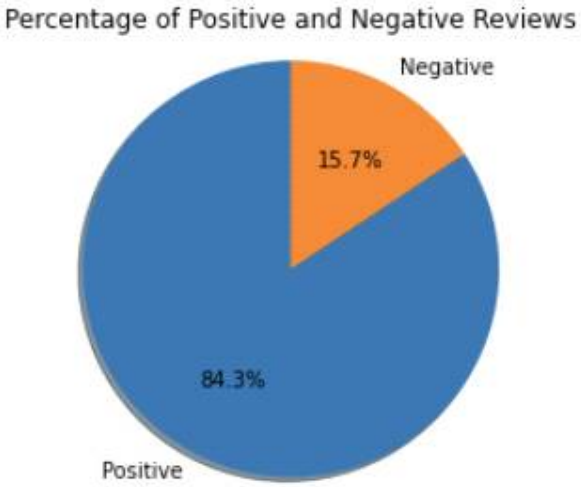
Time: 작성 시간

Summary: 리뷰 요약

Text: 리뷰 전문

만약 Score만을 기준으로 1, 2점을 준 리뷰는 'Negative', 4, 5점을 준 리뷰는 'Positive'로 분류한다면, 원자료 중 약 84.3%가 긍정적임을 알 수 있다. 하지만, 단순히 별점만으로 텍스트의 감성을 판단하는 것은 다소 naive 하므로, 이 결과는 대략적인 데이터의 성향을 알아보는 정도로 참고하겠다. 이에 관한 그래프는 다음 장에 첨부하였다.





이로써 이후 도출될 목적변수 또한 상당한 불균형(Imbalanced) 구조를 가질 것을 예상하여, 각 모형의 분류 성능을 판단하는 데 더욱 유의해야 할 것이다. 한편, 원 데이터는 다음과 같이 총 568411개의 관측치를 포함한다.

```
df.shape
```

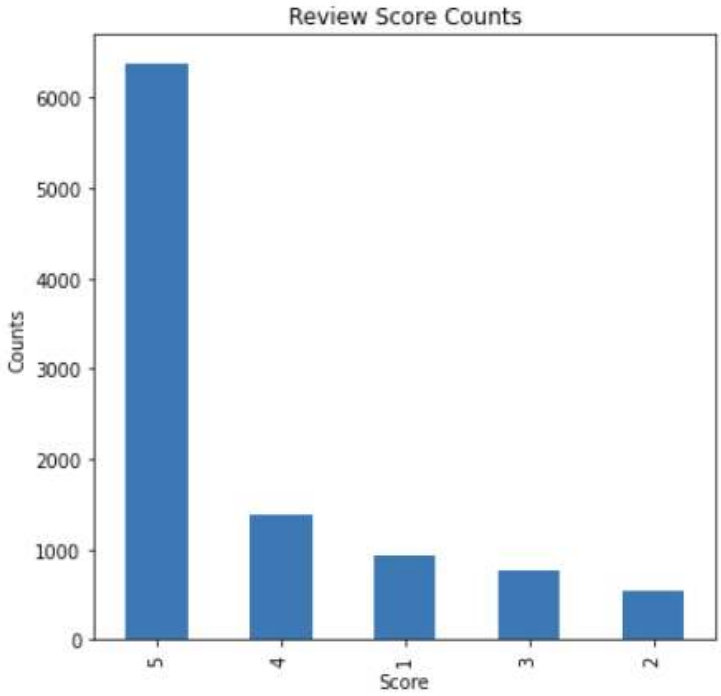
```
(568411, 9)
```

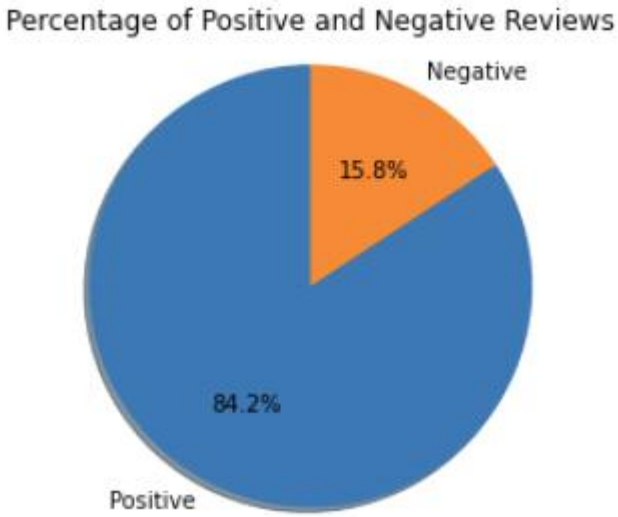
이 중 본 프로젝트는 다음과 같이 10000개의 표본을 임의로 추출하고, 결과의 일치성을 보장하기 위해 총 10번의 시뮬레이션을 할 것이다.

```
data =df.sample(n=10000, random_state=1)
data.shape
```

```
(10000, 9)
```

아래 두 표를 참고하면, 임의의 해당 표본은 앞선 원자료의 분포와 매우 유사한 패턴을 지닌다. 전체의 약 84%가 긍정적인 별점을 주었으므로, 분류 모형의 성능이 84%보단 높게 나오는 것이 바람직할 것이다. 이는 모든 데이터에 대해서 긍정적이라고 분류했을 때도 약 84% 정도의 정확도가 나올 가능성이 있기 때문이다.





1.4. 모델 평가 지표

모형 비교에 사용할 성능 평가 지표는 정확도(Accuracy)와 데이터 구조가 불균형일 때 주로 사용되는 F1-값이다. 이때 정확도의 경우, nested cross-validation에 의해 구한 Test 데이터의 평균 Accuracy로 도출하되, Train 데이터의 정확도와 비교하여 과대 적합 혹은 과소 적합의 여부를 판단할 것이다. 아래의 표를 참고하여, 각 지표에 대한 부가설명을 하겠다.

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

1.4.1. 정확도

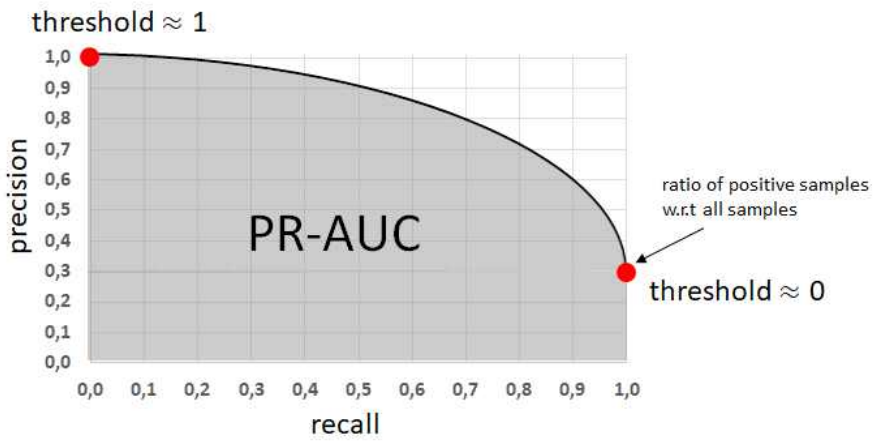
가장 흔히 쓰이는 모델 성능 평가 지표로, Accuracy 라고도 불린다. 이는 실제값과 예측치가 일치한 정도를 나타내며, $Accuracy = \frac{TP+TN}{TP+FN+FP+TN}$ 로 구해진다. 하지만, 주어진 데이터가 불균형자료일 때, 정확도만으로 모델을 평가하는 것은 바람직하지 않다. 앞서 언급했듯, 본 프로젝트의 데이터를 적합했을 때 모든 경우에 대하여 무조건 '긍정'이라 분류한다면 약 84%의 정확도가 나올 수 있기 때문이다.

1.4.2. F1-값

F1-값을 설명하기 위해, 우선 정밀도(Precision)와 재현율(Recall)에 대한

정의를 내려야 한다. 정밀도는 $\mathrm{Precision} = \frac{TP}{TP+FP}$로 계산되며, 예측값이 'True'인 것 중 실제로도 'True'인 정도를 나타낸다. 반면 재현율의 경우, $Recall = \frac{TP}{TP+FN}$로 계산되며, 실제로 'True'인 것 중 예측 값도 'True'인 정도를 나타낸다. 이처럼 정밀도와 재현율은 상충관계에 있기 때문에, 항상 두 지표를 함께 평가해야 한다. 둘 중 하나가 극단적으로 큰 경우보단 적당한 조화를 이루는 경우가 더 좋기 때문이다. 이를 나타내는 지표가 정밀도와 재현율의 조화평균인 F1-값이다. $F1\,Score = 2 \times \frac{1}{\frac{1}{\mathrm{Precision}} + \frac{1}{Recall}}$로 계산되며, 특히 불균형 데이터에 대한 모델 성능평가에 유용하다. F1-값은 각 범주별 관측치 개수를 고려하는 micro 방식과 이를 고려하지 않는 macro 방식을 모두 사용할 것이다.

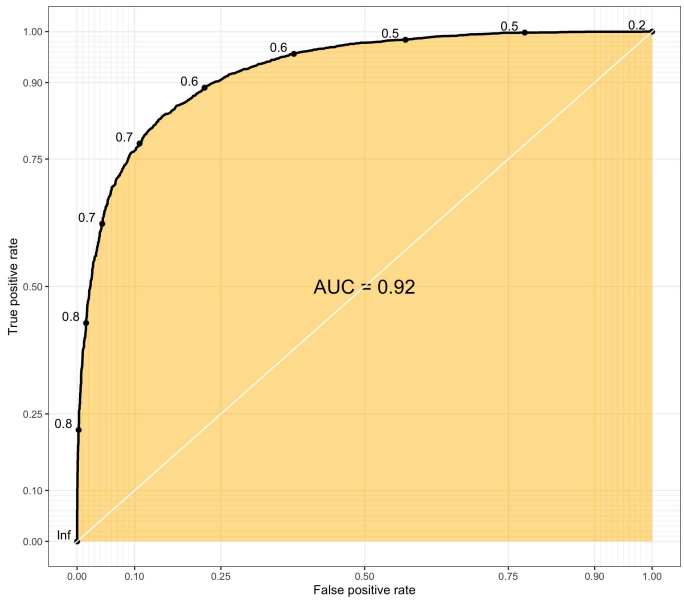
1.4.3. PR-AUC와 ROC-AUC



앞서 언급한 재현율과 정밀도를 각 축에 놓고 Precision-recall curve를 그렸을 때, PR-AUC은 Precision-recall curve 아래의 면적으로, 위 그래프에서 색칠된 부분의 면적이다. 이 또한 불균형 데이터를 다룰 때 모델의 평가 척도로 자주 활용되며, PR-AUC의 면적이 클수록 예측 성능이 더 좋은 모델이라고 판단한다. 소수 클래스의 분류 성능보단, 다수 클래스(+) 혹은 데이터 전반을

분류하는 성능을 중요시할 때 사용한다.

이와 달리, 아래 그림처럼 FPR(False Positive Rate)과 TPR(True Positive Rate)을 각 축에 놓고 ROC curve를 그렸을 때, 곡선 아래 면적을 ROC-AUC라고 한다. 앞서 언급한 재현율과 정밀도와는 조금 다른 개념으로, TPR과 FPR은 각각 $TPR = \frac{TP}{TP + FN}$, $FPR = \frac{FP}{TN + FP}$으로 계산된다. 이는 PR-AUC와 달리 데이터의 불균형을 무시하고, 소수 클래스(혹은 -)의 분류 성능 또한 중시할 때 사용한다. 본 프로젝트에선 이 두 지표를 모두 사용하여 모델을 평가하고, 각 상황에 따라 어떤 분류 모델이 가장 유용할지 판단하겠다.



제 2장. 데이터 분석

2.1. 텍스트 전처리

2장은 총 10개의 시뮬레이션 데이터 중 첫 번째 사례를 통해 분석 과정을 설명한다. 먼저 10000개의 표본에서 작성자 ID, 이름, 시간, 그리고 리뷰 전문이 중복되는 데이터를 삭제하였다. 그 결과, 총 9850개의 관측치로 표본의 크기가 줄었다.

```
data.drop_duplicates(subset={'UserId', 'ProfileName', 'Time', 'Text'}, inplace=True)
data.shape
```

```
(9850, 9)
```

다음으로, 필요없는 열을 지우고 아래와 같이 'Summary'와 'Text'열에 대해 텍스트 전처리를 진행하였다. 자세한 코드는 부록을 참고하길 바란다.

1) html 태그 삭제
2) 교정부호와 특수문자/기호 삭제
3) 영어/숫자가 아닌 단어 삭제
4) 모든 단어를 소문자로 바꾸기
5) 불용어(Stopwords) 삭제
6) 두 글자 이하인 단어 삭제

※불용어란, 관사, 전치사, 조사, 접속사 등 자연어 처리에 있어 의미를 거의 갖지 못하는 단어를 말한다.

아래는 위 전처리 과정을 거친 텍스트 데이터의 결과이다. 불필요한 어구가 제거되고, 핵심 명사 및 형용사 위주로 내용이 추출된 것을 볼 수 있다.

```
data.drop(['Id'], axis=1, inplace=True)
data.head()
```

	Score	Summary	Text
0	5	love dark chocolate want variety try	saw product best little interesting grocery st...
1	4	delicious	honey really good dissolves well hot drinks fo...
2	4	best protein shake ever	lean body chocolate ice cream ounces fantastic...
3	5	crumbs odor danger fun great hardcore chewers ...	think himalayan dog chews coolest thing sincer...
4	5	wonderful	better even expected opened whole new field ca...

이제 각 'Summary'열과 'Text'열에 대해 워드 클라우드(Word Cloud)를 만들어보았다. 이는 글에서 언급된 핵심 단어를 시각화하는 기법으로, 크게 표현될수록 많이 언급된 단어라는 뜻이다. 리뷰 별점을 기준으로, 이를 통해 낮은 점수를 준 고객과 높은 점수를 준 고객의 전반적인 단어 사용 차이를 볼 것이다. 그 결과는 다음과 같다.

먼저, 4, 5점을 준 고객의 리뷰 요약(Summary)에 대한 워드 클라우드이다. 긍정적인 리뷰인 만큼, 'great,' 혹은 'good'등 긍정적인 단어들이 가장 많이 쓰였다.

다음으로, 동일한 고객들의 리뷰 전문(Text)에 대한 워드 클라우드이다. 이 또한 'good'와 같은 긍정적인 단어들이 보이나, 'bad,' 'hate,' 'disappointed'와 같은 부정적인 단어 또한 함께 보이는 것으로 보아, 앞서 언급했던 단순히 별점만으로 글

쓴이의 감성을 판단할 수 없다는 주장의 근거가 된다. 예컨대, 별점은 5점을 주었어도, 전문에는 좋지 않은 감정을 내포했을 수 있다. 즉, 리뷰 내용과 별점이 일치하지 않는 경우를 고려하지 못한다.

다음은 아래 1, 2점을 준 고객의 두 워드 클라우드이다. 차례대로, 요약된 리뷰와 리뷰 전문에 대한 워드 클라우드이다. 앞서 지적했던 것처럼, 이 경우 'not good' 등의 어구에 의해 'good'를 뽑아내는 것을 방지하기 위해 다수의 긍정적인 단어를 불용어로 처리해주었다.

이 경우, 긍정적인 점수를 준 고객에 비해, 대체로 확실하게 부정적인 감정을 가진 고객들이 낮은 점수를 줄 것이다. 하지만, 점수만으로 감성을 판단하는 것은 또 다른 한계가 있다. 바로 3점을 준 고객을 긍정/부정으로 분류하기가 어렵다는 것이

다. 따라서, 부정적인 점수를 준 고객의 감성은 '부정적'으로 분류하되, 나머지 리뷰에 한해선 대표적인 감성사전인 VADER을 활용할 것이다.

2.2. VADER

VADER(Valence Aware Dictionary and sEntiment Reasoner)는 대표적으로 검증된 감성 분석용 파이썬 라이브러리로, 집단지성을 통해 구축한 감성 사전과 더불어 통사적인 언어적 규칙을 추가로 활용하기 때문에, 다른 감정 사전보다 높은 정확도를 자랑한다. 특히 SNS 분야의 텍스트에 대해 특화되어, 이 경우 사람보다 높은 수준으로 감정을 인식하는 것으로 유명하다.

해당 패키지는 감정을 나타내는 단어에 각 'Positive,' 'neutral,' 'negative' 점수를 부여하여 이를 고려한 최종 compound score를 산출한다. 그렇게 도출된 점수가 0.05점 이상이라면 '긍정적', -0.05점 이하라면 '부정적'으로 분류한다. 아래는 주어진 데이터의 리뷰 전문('Text'열)에 VADER 분석기를 적용해본 결과이다.

```
analyzer = SentimentIntensityAnalyzer()
sent=[]
for row in data['Text']:
    vs=analyzer.polarity_scores(row)
    sent.append(vs)
sent=pd.DataFrame(sent)
sent.head()
```

	neg	neu	pos	compound
0	0.000	0.488	0.512	0.9898
1	0.000	0.494	0.506	0.9366
2	0.034	0.611	0.355	0.9638
3	0.083	0.557	0.360	0.9981
4	0.000	0.734	0.266	0.4404

위 분류 기준에 따라 각 리뷰를 분류한 결과, 기존 별점을 기준으로 1466명이 '부정적'으로 분류되었다면, VADAR 적용 이후론 그 수가 1700명으로 늘어난 것을 확인할 수 있다.

```
data.Sentiment[np.where(data['Score']<3)[0]]=0
```

```
len(np.where(data['Sentiment']==0)[0])
```

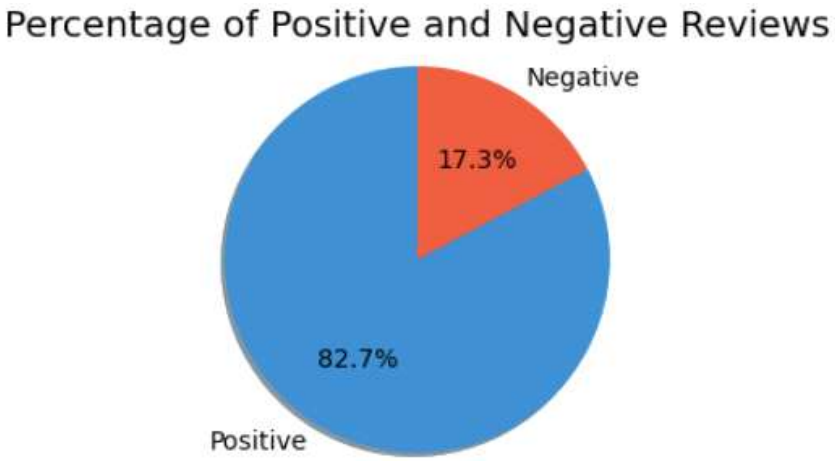
```
1700
```

```
len(np.where(data['Score']<3)[0])
```

```
1466
```

앞서 별점 3점을 준 고객을 제외한 naive한 방식으로는 약 15.7%의 고객이 '부

정적'으로 분류되었다. 이제 리뷰 내용과 별점이 일치하지 않는 고객을 재분류하고, 별점 3점을 준 고객을 추가로로 분류한 결과, 전체의 약 17.3%의 고객이 '부정적'으로 분류되었음을 알 수 있다. 이 결과를 다시 시각화 해보면, 아래와 같다. 데이터의 약 82.7%가 '긍정'이므로, 예측 모델의 성능은 적어도 82.7%보단 좋아야 할 것이다. 모든 데이터를 '긍정'이라 분류해도, 해당 모델의 정확도는 82.7%가 될 것이기 때문이다.



이제 아래와 같이, 'Sentiment'라는 열 아래 각 텍스트에 대한 감성을 양/음(1/0)으로 분류한 레이블이 생겼다. 이는 이후 예측모형을 학습시키고, 성능을 파악하기 위한 데이터의 목적변수 y로 활용될 것이다.

```
data.head()
```

	Score	Summary	Text	compound	Sentiment
0	5	best chips ever	stumbled multigrain chips little store work lo...	0.9191	1
1	4	pretty good gluten free dairy free option	surprised find gf dairy free bar starbucks tra...	0.9670	1
2	5	good healthy breakfast snack	love fair amount protein sugar source directly...	0.9382	1
3	5	best litter ever smell	using litter years one cat one bag lasts entir...	0.6369	1
4	5	love	far favorite pretzels ever taste mild like pre...	0.9153	1

2.3. Bag of Words

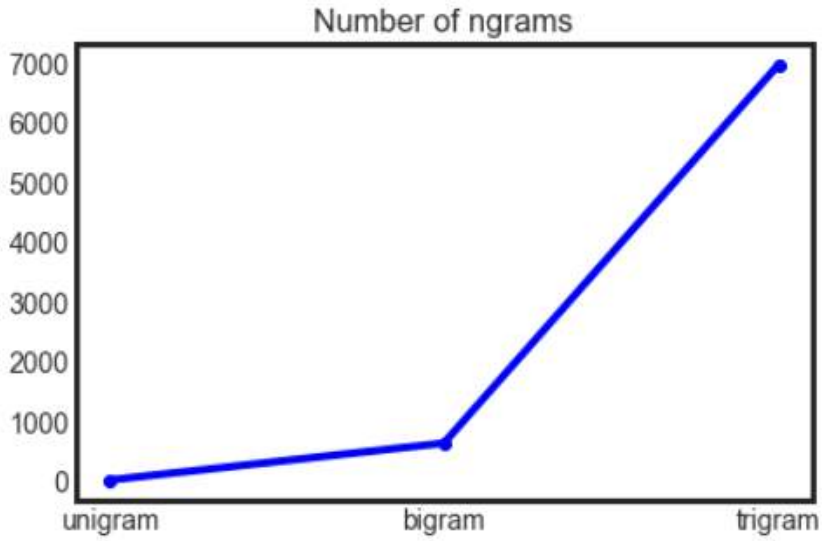
이제 텍스트를 수치형 피처 벡터로 표현해야 한다. 이때 가장 흔히 사용하는 모

델 중 하나가 바로 Bag-of-Words이다.

$$BoW(w,d) = number\ of\ times\ word\ w\ appears\ in\ document\ d$$

Bag-of-Words를 만드는 과정은 단순하다. 먼저, 전체 문서의 각 단어에 고유한 정수 인덱스를 부여한 뒤, 각 인덱스의 위치에 해당 단어가 각 문서 혹은 문장(corpus)마다 얼마나 자주 사용되었는지 횟수를 기록한 피처 벡터를 만드는 것이다. 이는 단어의 빈도수를 기반으로 문서를 표현하며, 단어의 순서를 전혀 고려하지 않은 채 오직 단어들의 출현 빈도에만 집중한다는 특징이 있다. 특정한 단어가 문서에 자주 등장할수록 중요하다고 판단하기 때문에, 불용어를 사전에 제거하는 것이 굉장히 중요하다.

이때, 추출하고자 하는 n개 이하의 연속된 단어 그룹(token)을 먼저 정의해야 한다. 이를 n-gram이라 하는데, 예를 들어 “Korea”과 “Korea University”는 2-gram에 속하지만, “Korea University in Seoul”는 속하지 못한다. 이렇듯 순서와 무관하게 n-gram으로 만들어진 토큰의 집합을 Bag-of-Words라고 하는 것이다. 원칙상 n-gram에서 n은 교차검증(cross-validation)을 통해 결정되어야 할 초모수이다. 하지만, 앞선 전처리를 거친 텍스트 데이터에 적용시 Unigram의 경우 27개, Bigram 645개, Trigram의 경우 6984개의 토큰이 생성되기 때문에, 본 프로젝트에선 computation의 한계 상 Unigram을 사용하겠다. 아래는 해당 사항을 그래프로 시각화한 것이다.



다음은 해당 Unigram Bag-of-Words 모델을 적용한 결과이다. 이처럼 여러 문서가 존재할 때, 각 문서에 대한 BoW 결과를 쌓아 행렬로 만든 것을 문서 단어 행렬(Document-Term Matrix)이라고 한다. 아래 행렬의 각 열이 이후 모델에 적합할 데이터의 설명변수 X로 입력되는 것이다.

```
df_bow = pd.DataFrame(x.todense())
df_bow.head()
```

	0	1	2	3	4	5	6	7	8	9	…	17	18	19	20	21	22	23	24	25	26
0	23	7	3	5	5	15	1	4	2	9	…	0	7	7	10	5	3	4	2	1	1
1	31	16	3	5	10	21	5	8	7	9	…	0	13	12	14	4	2	2	0	3	1
2	30	14	1	5	7	20	5	8	2	15	…	0	16	13	11	6	5	1	0	2	0
3	25	10	4	4	5	17	2	3	2	9	…	0	8	9	16	2	1	0	1	3	0
4	36	18	1	8	3	25	3	3	11	16	…	0	15	14	22	5	4	2	1	8	3

비교를 위해 원 데이터의 문서 단어 행렬 또한 첨부했다.

```
df_bow = pd.DataFrame(x.todense())
df_bow.head()
```

	0	1	2	3	4	5	6	7	8	9	…	102	103	104	105	106	107	108	109	110	111
0	0	0	0	0	0	0	48	0	0	0	…	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	30	0	2	0	…	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	98	0	2	0	…	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	42	0	0	0	…	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	29	0	0	0	…	0	0	0	0	0	0	0	0	0	0

이처럼 전처리 및 데이터 클리닝 작업을 하기 전의 경우, 피처 벡터 대부분의 값이 0이다. 각 문서의 단어들은 해당 문서 내에서의 부분집합을 나타낼 뿐이기 때문에, 다른 문서에서 동일한 단어가 있을 확률이 그리 높지 않기 때문이다. 이렇게 대부분의 값이 0인 벡터를 희소 벡터(sparse vector)라고 하는데, 이처럼 문서의 개수가 많음에 따라 단어 벡터의 크기만 매우 커지고, 대부분의 값이 0이라면 이는 시·공간적 낭비가 될 것이다. 따라서 텍스트 전처리 방법을 사용하여 빈도수가 낮은 단어, 불용어를 제거하여 단어 집합의 크기를 줄이는 과정이 필수적이다. 이렇듯 비슷하게 자주 등장하는 불용어와 중요한 단어 간 가중치를 조절하지 못한다는 것이 Bag-of-Words 모델의 한계이다.

2.4. TF-IDF

Bag-of-Words 못지않게 텍스트를 수치형 피처 벡터로 표현하는 데 자주 쓰이는 모델이 바로 TF-IDF(Term Frequency-Inverse Document Frequency)이다. 이는 앞선 BoW의 한계를 극복해, 단어의 빈도와 역빈도를 함께 고려한다는 점에서 BoW와 구분된다. 즉, 문서 단어 행렬(DTM) 내 각 단어들의 중요도에 가중치를 줄 수 있다. TF-IDF는 TF(Term Frequency)와 IDF(Inverse Document Frequency)를 곱한 값이며, $tf(d,t)$는 앞선 $BoW(w,d)$와 같다. 바로, 특정 문서 d에서 특정 단어 t의 등장 횟수를 구한 값이다. IDF는 다음과 같이 구해진다.

$$idf(d,t) = \log(\frac{n}{1+df(t)})$$

이때 $df(t)$는 document frequency로, 특정단어 t가 등장한 문서의 수를 뜻한다. 즉, 한 문서에서 특정 단어가 얼마나 사용되느냐가 아닌, 오직 등장했냐, 안 했냐를 따지는 함수이다. 분모에 1을 더하는 것은 $df(t)$가 0인 경우를 방지하기 위해서이며, 문서의 개수가 커질수록 IDF 값이 기하급수적으로 커지는 것을 방지하기 위해 자연로그를 취한다.

TF-IDF는 모든 문서에서 자주 등장하는 단어의 중요도는 낮고, 되려 특정 문서에서만 자주 등장하는 단어의 중요도가 높다고 판단한다. TF는 특정 문서에서 한 단어가 얼마나 자주 등장했는지를, IDF는 해당 단어가 전체 문서에서 얼마나 적게 등장했는지를 나타내기 때문이다. 만약 특정 문서에서 자주 등장했지만, 여러 문서에서도 자주 등장했다면, IDF에 의해 해당 단어의 가중치는 낮아질 것이다. 이러한 속성 때문에 TF-IDF는 불용어 처리에 아주 효과적이다. 하지만, TF-IDF가 BoW보다 항상 좋은 성능을 가지는 것은 아니다.

동일한 TF를 기반으로 하기 때문에, 앞서 만들어 놓은 DTM을 그대로 사용하여 TF-IDF 행렬로 변환할 수 있다. 해당 결과는 다음 페이지에 첨부하였다. 보다 다양한 비교를 위해, 아래 행렬과 앞선 BoW의 변수 행렬 모두 모델에 적합해 볼 것이다.

```
tfidf = text.TfidfTransformer(norm=None)
X_tfidf = tfidf.fit_transform(x)
df_tfidf = pd.DataFrame(X_tfidf.todense())
df_tfidf.head()
```

	0	1	2	3	4	5	6	7	8	9	...	17	18	19	20	21	22	23	24
0	23.060785	7.040621	3.429139	5.111891	5.122803	15.045750	1.124825	4.148916	2.157499	9.084447	...	0.0	7.042051	7.041336	10.054968	5.212026	3.489116	4.803134	3.853193
1	31.081927	16.092848	3.429139	5.111891	10.245607	21.064050	5.624124	8.297832	7.551245	9.084447	...	0.0	13.078094	12.070862	14.076955	4.169621	2.326077	2.401567	0.000000
2	30.079284	14.081242	1.143046	5.111891	7.171925	20.061000	5.624124	8.297832	2.157499	15.140746	...	0.0	16.096116	13.076767	11.060464	6.254431	5.815193	1.200783	0.000000
3	25.066070	10.058030	4.572185	4.089513	5.122803	17.051850	2.249650	3.111687	2.157499	9.084447	...	0.0	8.048058	9.053146	16.087948	2.084810	1.163039	0.000000	1.926596
4	36.095141	18.104454	1.143046	8.179026	3.073682	25.076251	3.374474	3.111687	11.866242	16.150129	...	0.0	15.090109	14.082672	22.120929	5.212026	4.652154	2.401567	1.926596

2.5. 모형 구축

본 프로젝트에서 비교할 학습 모형은 로지스틱 회귀(Logistic Regression), 서포트 벡터 머신(SVM), 의사 결정 나무(CART), 그리고 랜덤 포레스트(Random Forest)이다. 각 모형의 초모수(hyperparameter)는 각자 5-fold 교차검증(Cross Validation)의 그리드 서치(GridSearch)를 통해 선택할 것이다. 또한, 모델의 학습과 성능평가를 위해 앞선 두 가지 데이터(X: BoW, TF-IDF, y는 동일)를 각각 7:3의 Train 데이터와 Test 데이터로 나눌 것이다. 이때 X의 경우, 총 27개의 설명변수로 이루어져있으므로, 주성분분석(Principal Component Analysis)의 팔꿈치 방법(Elbow Method)에 의해 10개의 주성분으로 차원 축소할 것이다. 다음으로, 목적변수 y가 굉장한 불균형 데이터이므로, 앞서 정의한 모델 평가 지표 중 정확도를 제외하고, F1-값과 PR-AUC, 그리고 ROC-AUC만으로 모형의 분류 성능을 판단할 것이다. 전체적인 데이터에 대한 성능도 중요하지만, 소수 클래스인 '부정'을 잘 분류할 수 있는 지도 중요한 평가 요소가 될 것이므로, micro와 macro F1값 및 PR-AUC와 ROC-AUC간 특별한 우위 없이, 전반적인 수치로 평가할 것이다.

2.5.1. 로지스틱 회귀

로지스틱 회귀 분석은 반응변수가 범주형 변수일 때 가장 흔히 쓰이는 분석 모형이다. 본 아마존 리뷰 데이터의 경우, 앞서 각 리뷰의 별점과 VADER를 통해 각 문서의 감성을 '긍정(1)' 혹은 '부정(0)'으로 나타낸 이항 반응변수를 만들었으니, 해당 모형을 사용할 수 있다. 이 경우, 로짓 모형(logit model)을

사용한다.

여기서 반응변수 Y에 대한 추가적인 가정이 필요하다. 바로, Y는 설명변수 $X=x$일 때 성공 확률 p_x를 갖는 베르누이 분포를 따른다는 것이다. 이 가정 하에 $Y=y$일 확률은, $P(Y=y|X=x)=p_x^y(1-p_x)^{1-y} \quad where \quad y=0,1$가 된다. 즉, 이 경우 p_x는 문서의 감성이 '긍정(1)'일 확률을 뜻한다.

이때, 위 식의 p_x에 대해 $p_{x=}\alpha+\beta x$라는 선형 모형을 만들 수 있는데, 이는 $0 \le p_x \le 1$ 라는 구조적 결함을 가지고 있다. 따라서, α와 β에 대한 제약이 없는 다음과 같은 로짓 모형을 사용한다.

$$\ln\frac{p_x}{1-p_x}=\alpha+\beta X$$

이는 기본 오즈 모형인 $\frac{p_x}{1-p_x}=\exp(\alpha+\beta X)$에 로그를 씌운 것으로, log odds를 줄여 로짓(logit)이라고 하는 것이다.

위 로짓 모형에 앞서 정립한 두 가지 문서 벡터를 X로, 전처리된 데이터의 'Sentiment' 열을 y로 입력할 것이다. 이 벡터들을 바탕으로 학습된 로지스틱 회귀 모델은 새롭게 입력된 문서에 대해 이 문서의 감정이 '긍정'일 확률을 반환할 것이다. 이에 0.5를 기준 점(threshold)로 설정하여, 각 문서를 '긍정' 또는 '부정'으로 분류하기로 한다.

우선 Bag-of-Words 모델을 사용하여 변환한 X 변수를 10개의 주성분(Principal Component)으로 차원 축소(PCA) 후, 로짓 모형에 적합 해보았다. 그 결과, 세 변수의 회귀계수의 p-value가 유의수준인 0.05보다 작거나 매우 유사하으므로, 모든 변수에 대해 아래 귀무가설을 기각하였고, 설명변수가 모두 유의하다는 결론을 내렸다.

H_0 : 설명변수의 회귀계수 값은 0이다.
H_1 : 설명변수의 회귀계수 값은 0이 아니다.

각 설명변수의 회귀계수에 대한 자세한 결과 및 p-value는 다음장에 첨부된 그림을 참조하길 바란다.

Logit Regression Results

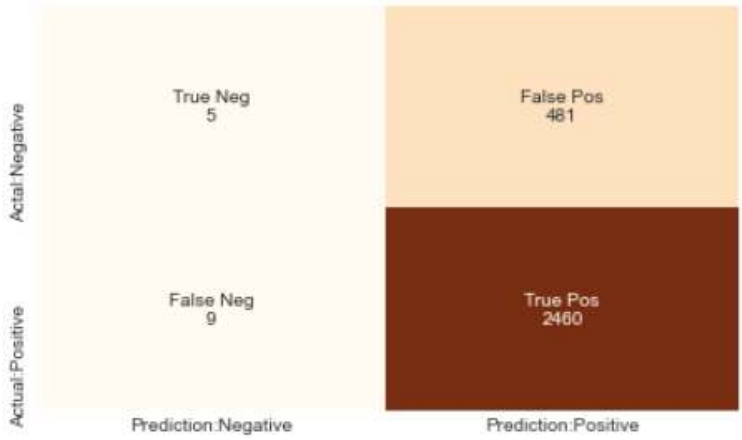
	coef	std err	z	P>|z|	[0.025	0.975]
0	-0.0545	0.011	-5.152	0.000	-0.075	-0.034
1	0.0154	0.008	2.029	0.042	0.001	0.030
2	0.0937	0.013	7.023	0.000	0.068	0.120
3	0.0623	0.014	4.602	0.000	0.036	0.089
4	0.0292	0.013	2.183	0.029	0.003	0.056
5	0.0678	0.017	3.928	0.000	0.034	0.102
6	-0.0077	0.009	-0.863	0.388	-0.025	0.010
7	0.0172	0.009	1.921	0.055	-0.000	0.035
8	0.0011	0.009	0.130	0.897	-0.016	0.018
9	0.0008	0.010	0.076	0.940	-0.019	0.021

<로짓 모형 적합 결과>

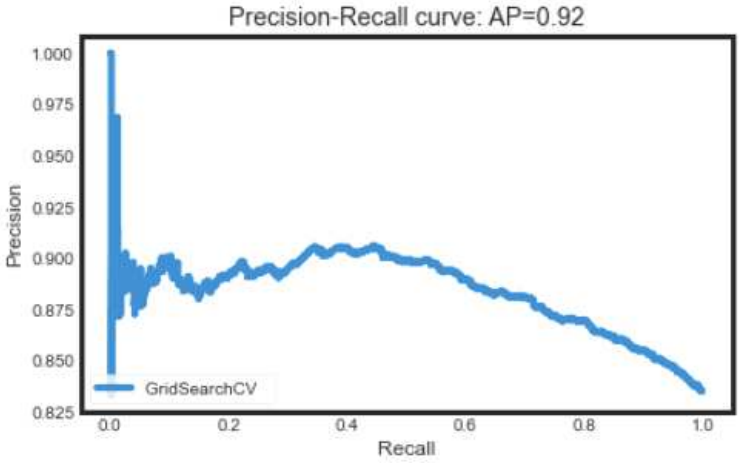
다음으로, 그리드 서치를 통해 최상의 초모수 $C=1/\lambda$는 0.001로 결정되었다. 또한, 모형의 과대적합(Over Fitting)을 방지하기 위해 L2 규제화(Regularization)인 릿지(Ridge) 페널티를 사용하였다. F1-값은 micro의 경우 0.84, macro의 경우 0.46, PR-AUC값은 0.92로, ROC-AUC값은 0.50으로 도출되었다. macro F1-값과 ROC-AUC값으로 보건데, 소수 클래스인 '부정'을 잘 분류하지 못한 것으로 판단된다. 실제로 오차행렬 (Confusion matrix)를 살펴보면, 총 486개의 실제 '부정' 중 오직 5개만을 정확히 '부정'이라고 분류했음을 알 수 있다. 결과는 아래를 참고하길 바란다.

```
{'C': 0.001}
Test:
 F1 score(micro) / F1 score(macro) / PR-AUC / ROC-AUC
 0.83418          / 0.46471    / 0.91793     / 0.50332
```

<교차 검증 및 모형 성능 결과>



<오차행렬>



<Precision-Recall Curve과 PR-AUC(AP)>

동일한 조건 아래 TF-IDF 모델로 변환한 X 변수를 적합해 보았을 때도, 비슷한 결과가 나왔다. 이전과 동일하게 세 변수의 회귀계수의 p-value가 유의수준 0.05보다 작아, 모든 경우에 대해 다음장에 명시된 귀무가설을 기각하고 해당 변수가 유의하다는 결론을 지었으며, 그리드 서치 결과 최상의 초모수 $C = 1/\lambda$ 또한 0.001로 같았다.

H_0 : 설명변수의 회귀계수 값은 0이다.
H_1 : 설명변수의 회귀계수 값은 0이 아니다.

Logit Regression Results

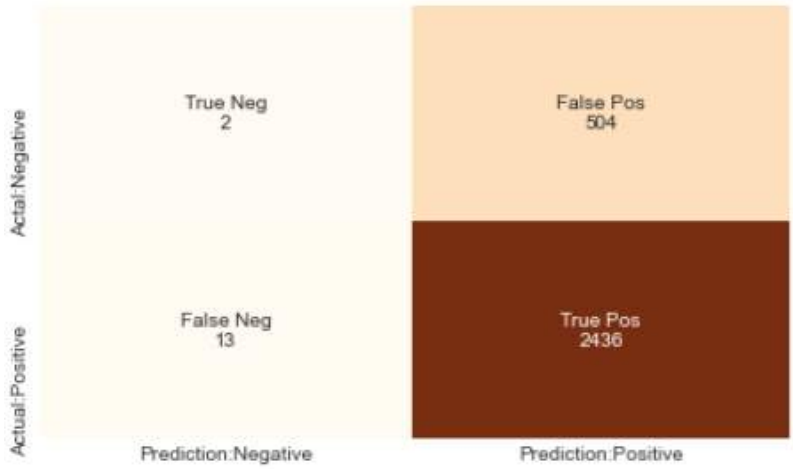
	coef	std err	z	P>|z|	[0.025	0.975]
0	-0.0244	0.011	-2.193	0.028	-0.046	-0.003
1	-0.0552	0.010	-5.332	0.000	-0.076	-0.035
2	0.0315	0.008	4.106	0.000	0.016	0.047
3	0.0922	0.012	7.635	0.000	0.068	0.116
4	0.0690	0.013	5.266	0.000	0.043	0.095
5	-0.0780	0.010	-8.088	0.000	-0.097	-0.059
6	0.0342	0.009	3.656	0.000	0.016	0.053
7	-0.0356	0.010	-3.514	0.000	-0.055	-0.016
8	0.1006	0.018	5.596	0.000	0.065	0.136
9	-0.0928	0.015	-6.159	0.000	-0.122	-0.063

<로짓 회귀 분석 결과>

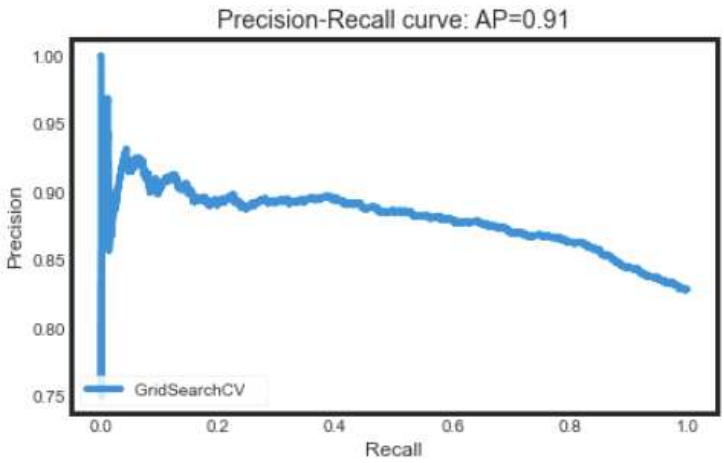
모델 성능면에선 F1-값은 micro의 경우 0.83, macro의 경우 0.45, PR-AUC 값은 0.91, ROC-AUC값은 0.50으로, 총 506개의 실제 '부정' 중 오직 2개만을 정확히 '부정'이라고 분류하여, 앞선 BoW의 경우보다 성능이 저하되었다. 이로써 TF-IDF가 BoW보다 결과가 항상 좋지만은 않다는 것을 증명할 수 있었다. 아래 그림에서 그 결과를 확인하길 바란다.

```
{'C': 0.001}
Test:
 F1 score(micro) / F1 score(macro) / PR-AUC / ROC-AUC
 0.82504          / 0.45587   / 0.91383     / 0.49932
```

<교차 검증 및 모형 성능 결과>



<오차 행렬>

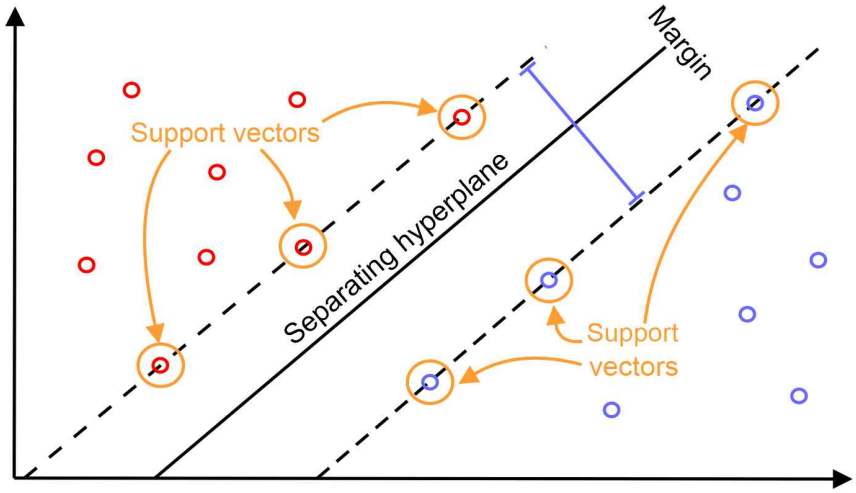


<Precision-Recall Curve과 PR-AUC>

2.5.2. 서포트 벡터 머신

서포트 벡터 머신(Support Vector Machine) 분류는 분류 규칙을 산출해내는 기계학습의 대표적인 지도학습 방법이다. n개의 개체들에게 p개의 속성(attribute) $X_1,\cdots,X_p$dl 측정되었고, 외적 정보로 Y가 -1 또는 1인 이항변수일 때, 어떤 속성의 개체들이 Y가 $+1$이 되는지 분류 규칙을 세우는 것이다.

SVM은 자료를 나누는 경계선 중 가장 폭이 넓은 것을 찾아낸다. 다음 장의 그림을 보자.



특정 자료를 (x_i^t,y_i) $(i=1,\cdots,n)$이라고 할 때, x_i는 $p\times1$ 설명 벡터, y_i는 -1 또는 +1값을 가지는 목적 벡터이다. 위 그림 속 찾고자 하는 선형 분류 경계면(hyperplane)을 $f(x)=w^tx+b$라고 하면, $f(x)>0$이면 y를 +1로, $f(x)<0$이면 y를 -1로 예측할 것이다. 이렇게 선형으로 분리되는 상황을 수리적으로 표현하면, 다음 두 가지 제약조건과 같다.

1) $w^tx_i+b\ \geq 1,\ (\text{if}\ y_i=1)$

2) $w^tx_i+b\ \leq -1,\ (\text{if}\ y_i=-1)$

즉, 위 그림에서 두 점선 간 폭은 $\|w\|$의 역수에 비례하게 된다. 따라서, SVM은 위 두 조건 아래 $\|w\|$를 최소화하는 것을 목표로 둔다.

$$\text{min}imize_w\ \frac{1}{2}\|w\|^2$$

이에 대한 해는 $w=\sum_{i=1}^{n}\lambda_iy_ix_i,\ \lambda_1,\cdots,\lambda_n\geq0$으로 구할 수 있으며, $\lambda_i>0$인 개

체 i의 설명 벡터 x_i를 서포트 벡터(support vector)라고 한다. 이들은 위 그림상 점선 위 점들처럼, $w^t x + b = \pm 1$의 경계에 놓인다. 하지만, 모든 데이터가 위처럼 완벽하게 나뉠 수 없을 것이다. 따라서, 아래와 같이 일부 개체들에 대하여 제약조건을 완화하고, 대신 패널티를 부과하는 방식이 보편적이다.

$$\mathrm{min}imize_w \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i, \quad (\xi_1 \geq 0, \cdots, \xi_n \geq 0)$$

$$subject\ to \quad 1)\ w^t x_i + b \geq 1 - \xi_i, (\text{if } y_i = 1), \quad 2)\ w^t x_i + b \leq -1 + \xi_i, (\text{if } y_i = -1)$$

여기서 ξ_i는 조건 완화를 위한 여분 변수(slack variable)이고, C는 이에 따른 패널티인 단위 비용(unit cost)이다. SVM은 경험적 위험 최소화 원칙(Empirical risk minimization)을 활용하여, 일반화 오류를 줄이기 때문에 다른 분류기법들과 비교하였을 때 우수한 성능을 보인다고 알려져 있다.

하지만, 대다수의 데이터는 선형적으로 분류되지 않을 것이다. 일반적으로 $p(\geq 3)$변량 자료의 경우, 그룹 분류를 위해 어떤 비선형 변환이 필요한지를 알아내기 쉽지 않다. SVM은 이를 해결하기 위해 설명공간을 힐버트 공간(Hilbert space)으로 옮긴다. p차원 설명벡터 x를 힐버트 공간의 $\Phi(x)$로 옮기면, Φ는 p차원 유클리드 공간 $\mathbb{R}^p$에서 힐버트 공간 $\mathbb{H}$로의 함수이다. 이 $\mathbb{H}$에서 두 개체 $\Phi(x_1)$과 $\Phi(x_2)$간 내적인 $\Phi(x_1) \bullet \Phi(x_2)$은 특정 커널 함수 $K(x_1, x_2)$로 얻어지는데, 본 프로젝트에선 그리드 서치를 통해 최상의 커널을 찾아낼 것이다. 후보군에 놓은 커널 함수는 다음과 같다.

1) 선형 커널(linear kernel, 'linear'): $K(X, Y) = X^T Y$

2) 가우스 커널(radial kernel, 'rbf'): $K(X, Y) = \exp(-\gamma\|x_1 - x_2\|^2), \ \gamma > 0$

3) 다항 커널(polynomial kernel, 'poly'): $K(X, Y) = (\gamma x_1^t x_2 + c_0)^d, \ d = 1, 2 \cdots$

4) 로지스틱 커널(sigmoid kernel, 'sigmoid'):

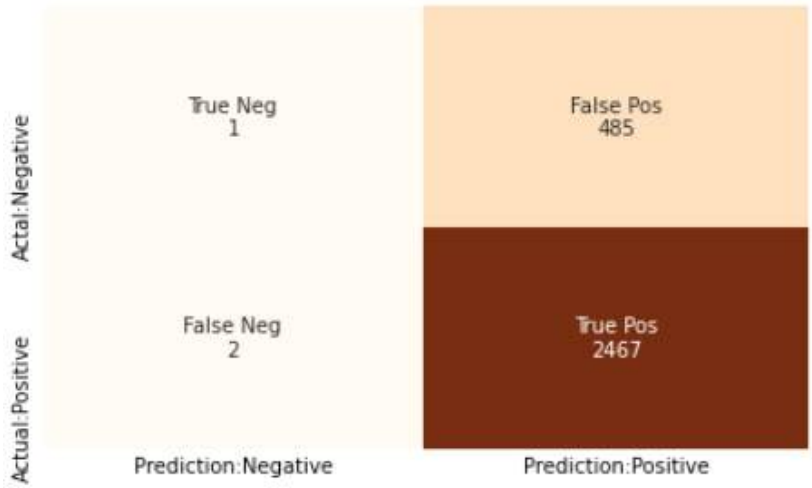
$K(X, Y) = \tanh(\gamma x_1^t x_2 + c_0), \ c_0 \geq 0, \gamma > 0$

이러한 SVM을 감성 분석에 적용하자면, 이는 '긍정'과 '부정'인 문서들을 가려내는 일종의 분류 경계면(hyperplane)을 찾는 것을 목표로 둔다.

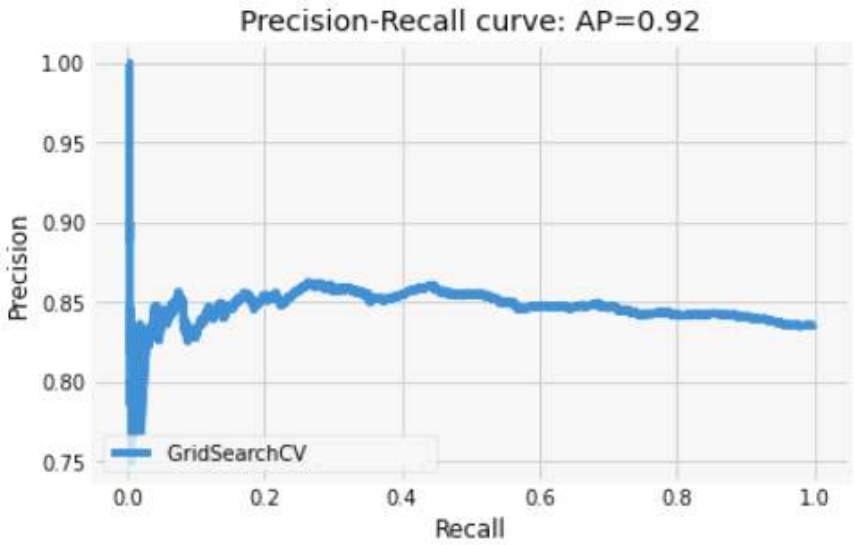
우선 Bag-of-Words 모델을 사용하여 변환한 X 변수를 모형에 적합 해보았

다. 이 경우 최상의 초모수로 C는 0.1, γ는 0.01, 그리고 커널 함수로는 가우시안 커널인 'rbf'가 설정되었다. F1-값은 각 micro와 macro에 대하여 0.84와 0.46, PR-AUC값은 0.92, ROC-AUC값은 0.5로, 앞선 로짓 모형의 결과와 거의 일치했다. 하지만 실제로 오차 행렬을 분석한 결과, 거의 모든 데이터에 대해 '긍정'이라고 분류한 것을 확인할 수 있다. 실제로 '부정'인 총 486건 중 단 한 건만 제대로 분류하였다. 그 결과는 아래에서 확인할 수 있다.

```
{'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf'}
Test:
 F1 score(micro) / F1 score(macro) / PR-AUC / ROC-AUC
 0.83553          / 0.4552   / 0.91777     / 0.5
```



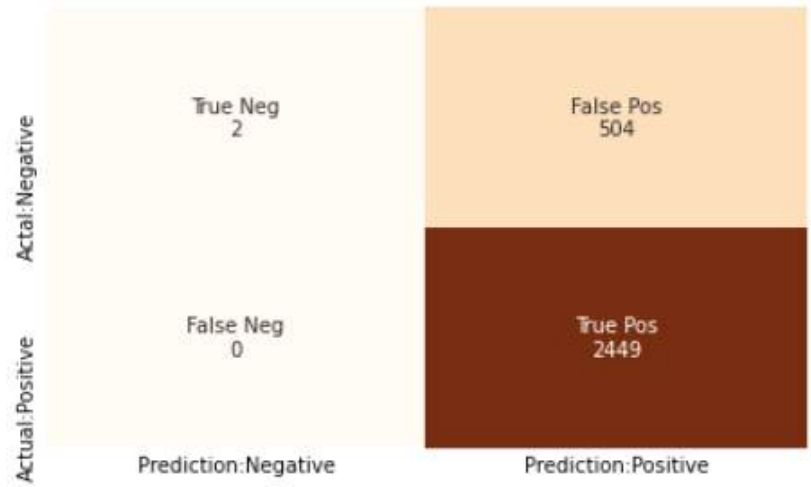
<오차 행렬>

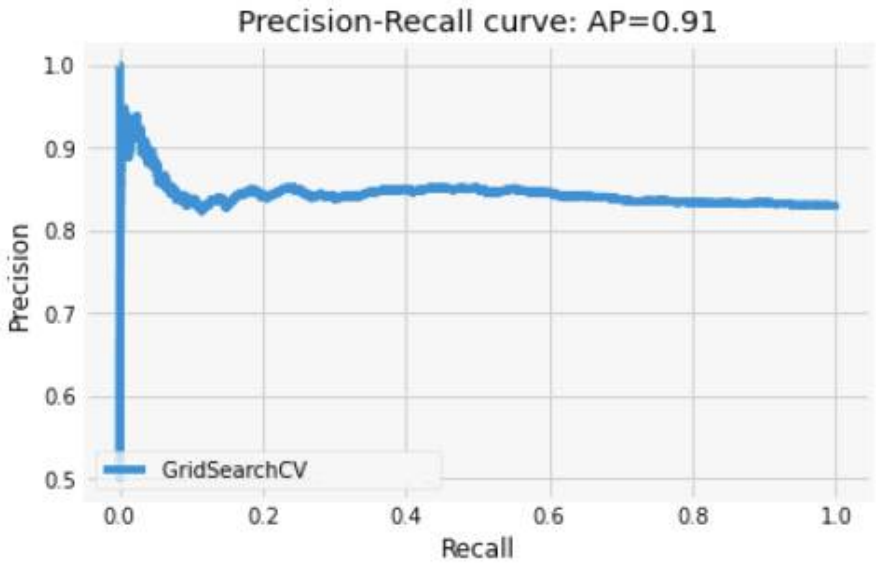


<Precision-Recall Curve과 PR-AUC>

동일한 조건 아래 TF-IDF 모델로 변환한 X 변수를 적합해 보았을 때도, 비슷한 결과가 나왔다. 그리드 서치 결과 최상의 초모수로 C는 0.1, γ는 0.01, 그리고 커널 함수로는 가우시안 커널인 'rbf'가 설정되었다. F1-값과 PR-AUC, ROC-AUC값은 각 0.83, 0.45, 그리고 0.91과 0.5로, 로지스틱 회귀의 결과와 비슷했다. 즉, SVM 모델 역시 BoW 데이터가 조금 더 높은 성능을 보였다. 실제로 오차 행렬을 분석한 결과, 실제로 '긍정'인 데이터는 모두 올바르게 분류하였으나, 실제로 '부정'인 총 506개의 데이터 중에선 단 두 건만 제대로 분류하였다. 그 결과는 아래에서 확인할 수 있다

```
{'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf'}
Test:
 F1 score(micro) / F1 score(macro) / PR-AUC / ROC-AUC
 0.82876          / 0.45318    / 0.91438     / 0.5
```





2.5.3. 의사결정나무

의사결정나무(Classification And Regression Tree)는 종속변수 Y가 이항형이고, 예측변수 $X_1,\cdots,X_p$가 존재할 때, 노드 분리(notde splitting)의 원리를 이용한다.

노드 분리란, 분리 대상인 노드를 M, 혹은 어미 노드라고 할 때 M을 자녀 노드 C_1과 C_2로 나누는 것을 뜻한다. $X_1,\cdots,X_p$ 중 하나인 X_j와 특정 값 k_j를 선택하여, $x_j \le k_j$인 개체는 노드 C_1에, $x_j > k_j$인 개체는 노드 C_2에 넣는 것이다. 여기서 핵심 아이디어는 해당 변수 X_j와 특정 값 k_j를 찾는 데 있다.

이를 위한 대표적인 지표로, 지니 지수(Gini index)가 있다. 임의 노드 N에서 $Y=1$일 비율을 p, $Y=0$일 비율을 q라고 했을 때, 지니 지수는 $G(N)=pq$로 정의되며, 노드의 불순도(impurity)를 측정한다. 예를 들어 $Y=1$과 $Y=0$의 비율이 $1:1$이고 관측치 수가 100인 어미 노드 M에서 어떤 변수 X_j와 k_j에 의해 2개의 자녀 노드 C_1과 C_2에 각각 50개씩 나뉘었다고 생각해보자. 만약 노드 C_1에서 $Y=1$일 비율이 0.8, $Y=0$일 비율이 0.2이고, 노드 C_2에서 $Y=1$일 비율이 0.2, $Y=0$일 비율이 0.8이라면, 지니계수는 $G(M)=0.5\times0.5=0.25$, $G(C_1)=G(C_2)=0.8\times0.2=0.16$이고, C_1과 C_2의 불순도 평균은 0.16이 된다. 어미 노드가 두 개의 자녀 노드로 분리되면서 불순도가 감소한 것이다.

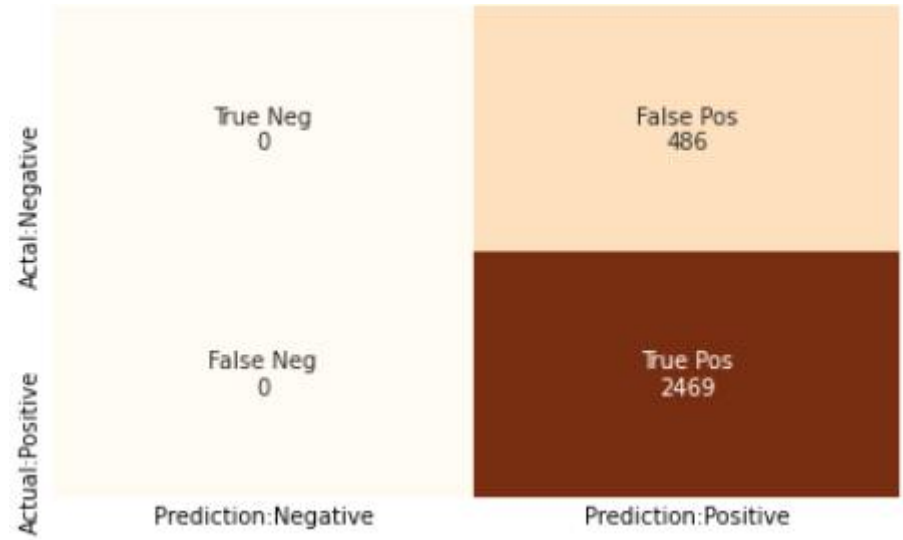
이렇듯, 노드 M의 분리는 $G(C_1), G(C_2)$의 가중 평균이 최소가 되도록 선택되며, 어떤 변수와 분리 값을 선택하더라도 불순도의 감소가 작은 경우, 해당 노드의 분리를 종결하게 된다.

또 다른 불순도 측도로, 교차 엔트로피 지수(Cross Entropy index)가 있다. 이는 $C(N)=-p\log_2(p)$로 정의되며, 지니 지수와 마찬가지로 노드 순도를 가장 증가시킨다고 판단하는 변수를 기준으로 노드를 나눈다. 본 프로젝트에선 그리드 서치를 통해 둘 중 더 좋은 결과를 도출하는 불순도 측도를 선택하고, 최상의 의사 결정 나무의 깊이(depth) 또한 이를 통해 설정해줄 것이다.

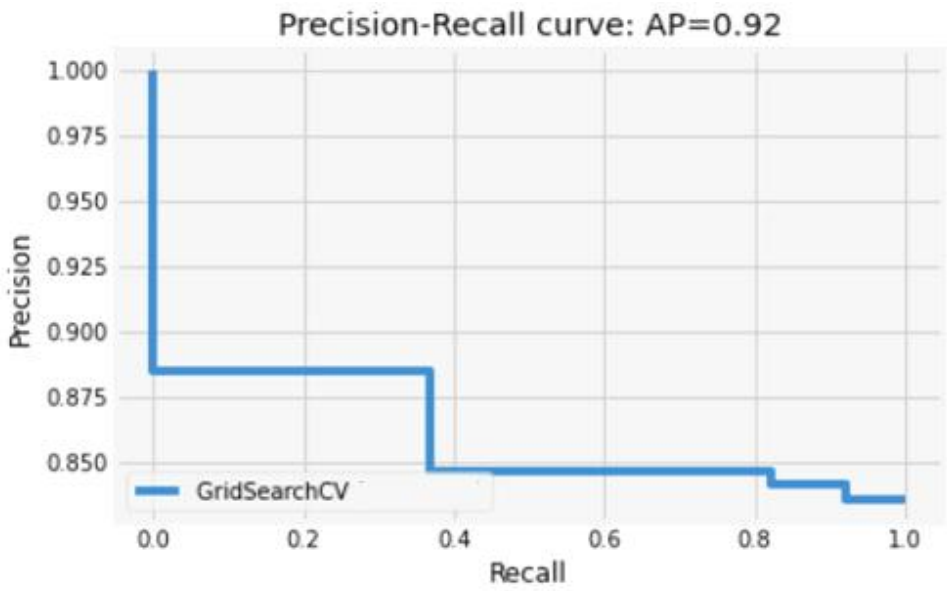
먼저 Bag-of-Words 모델을 사용하여 변환한 X 변수를 모형에 적합 해보았다. 이 경우 불순도 측도로 교차 엔트로피를, 최상의 깊이는 2로 설정해주었다. F1-값은 각 micro와 macro인 경우 0.84와 0.46, PR-AUC값은 0.92,

ROC-AUC값은 0.5로, 앞선 모형들과 유사해보였으나, 실제로 오차행렬을 분석한 결과, 모든 데이터를 '긍정'이라고 분류한 것을 확인할 수 있었다. 따라서, 마냥 성능이 괜찮다고 말할 수 없다고 판단하였다. 그 결과는 아래에서 확인할 수 있다.

```
{'criterion': 'entropy', 'max_depth': 2}
Test:
 F1 score(micro) / F1 score(macro) / PR-AUC / ROC-AUC
 0.83553          / 0.4552   / 0.91777     / 0.5
```



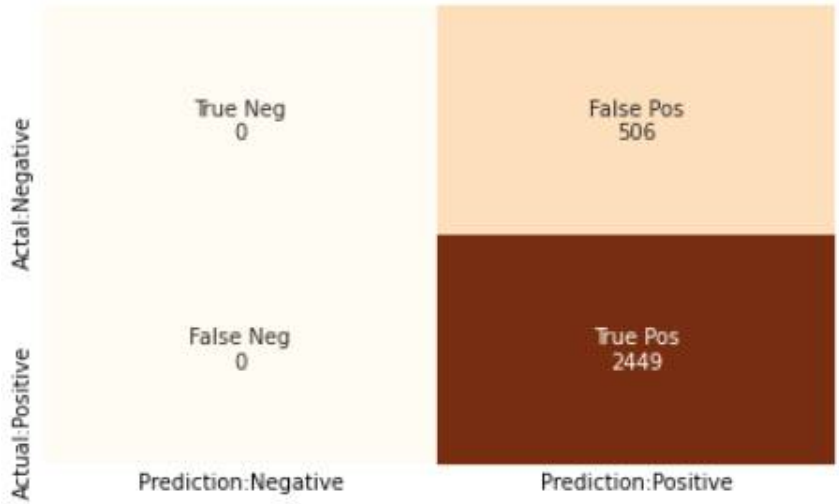
<오차 행렬>

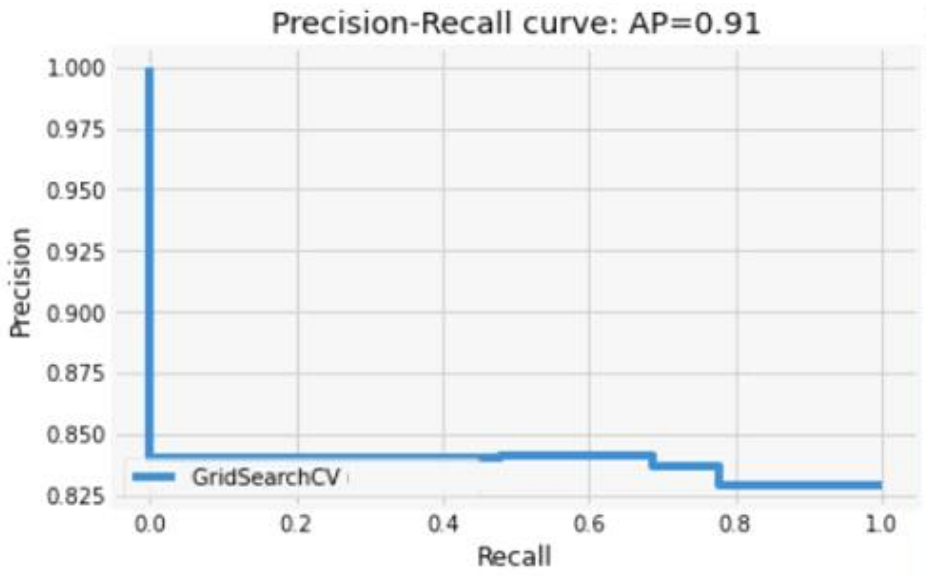


<Precision-Recall Curve & PR-AUC>

동일한 조건 아래 TF-IDF 모델로 변환한 X 변수를 적합해 보았을 때도, 비슷한 결과가 나왔다. 그리드 서치 결과 불순도 측도로 지니 지수를, 최상의 깊이는 앞선 경우와 같이 2로 설정해주었다. F1-값은 각 micro와 macro인 경우 0.83와 0.45, PR-AUC값은 0.91, ROC-AUC값은 0.5로, 이 또한 앞선 두 모델의 결과와 일치했으나, 실제로 오차 행렬을 분석한 결과, 마찬가지로 모든 데이터를 '긍정'으로 분류했음을 확인하였다. 그 결과는 아래에서 확인할 수 있다

```
{'criterion': 'gini', 'max_depth': 2}
Test:
 F1 score(micro) / F1 score(macro) / PR-AUC / ROC-AUC
 0.82876          / 0.45318    / 0.91438     / 0.5
```





2.5.3. 랜덤포레스트

앙상블 학습은 다양한 종류의 분류기법 및 회귀기법의 모임이며, 이를 통해 예측 성능을 높이는 기계학습 기법이다. 때문에, 앙상블기법은 대수의 법칙(Law of large number, LLN)에 기초한다. 즉, 독립적인 분류기법이 많을수록 분류의 정밀도가 높아진다. 그러나, 지금까지 개발된 분류기법의 수가 대수의 법칙을 적용할 많큼 많지는 않고, 각 기법 간 독립인지 검정할 수 있는 방법이 없기 때문에, 적은 수의 분류기법을 사용하되, 학습 데이터를 늘리는 방식이 발달하였다. 그중 대표적인 것이 Bagging인데, 랜덤포레스트(Random forest)는 바로 의사결정나무의 Bagging 버전이다.

Bagging은 학습 데이터의 크기가 n일 때, 중복을 허용하여(with replacement) 크기가 n인 M개의 새로운 학습 데이터 셋을 생성하는 방식이다. Bagging시 새로운 학습 데이터에 포함 되지 않는 데이터는 Oob(out-of-bag)라고 하며, 이는 시험(test) 데이터로 이용할 수 있다.

크기가 n이고 d개의 특성변수를 가진 원래 학습 데이터를 갖고 있다면, 랜덤포레스트의 분석 절차는 다음과 같다.

1) n개의 확률 붓스트랩 표본을 뽑아 새로운 학습데이터를 생성한다. 이때 중복을 허용하고, 뽑히지 않은 개체들은 테스트 표본으로 한다.

2) 새로운 학습 데이터를 의사 결정 나무 알고리즘에 투입하되, 각 노드에서 $q(< p)$개의 변수를 비복원 임의 추출하여 이 q개의 변수에 대하여 일반적인(최대한 큰) 의사결정 나무를 완성한다. 이때 q의 디폴트 값은 $\sqrt{p}$이다.

3) 위 절차를 총 M번 반복한다.

4) 산출된 M개의 의사 결정 나무 결과를 통합한다.

마지막 4번에서 의사 결정 나무 결과를 통합하기 위해선 투표분류기법(Voting classifier)을 사용한다. 앙상블 기법에서 투표 방법은 크게 두 가지로 나뉘는데, 먼저 Hard Voting의 경우, 결과마다 분류될 범주를 예측하여, 빈도가 가장 높은 범주로 최종 할당하는 방식이다. 이에 반해 Soft Voting의 경우, 모든 결과에 대해 각 범주에 속할 확률을 예측하고 평균 내어, 이 평균 확률이

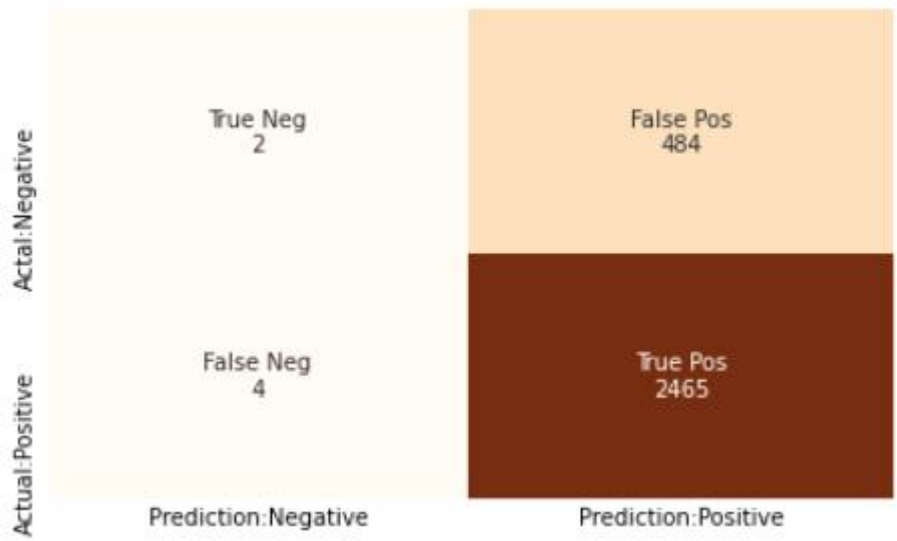
가장 큰 범주로 최종 할당하는 방식이다. 랜덤포레스트의 경우, Hard Voting의 일종인 Majority voting방식을 사용하여 다수결로 결과를 취합한다.

본 프로젝트에선 그리드 서치를 통해 앞선 분석 절차 3번의 이상적인 q값과, 앞선 의사 결정 나무와 동일하게 더 좋은 결과를 도출하는 불순도 측도를 선택할 것이다.

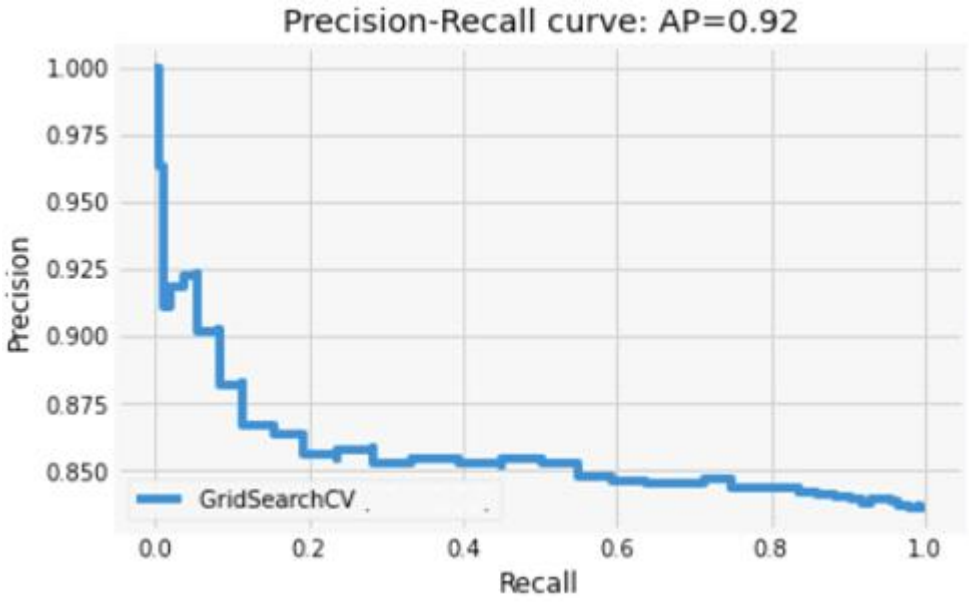
먼저 Bag-of-Words 모델을 사용하여 변환한 X 변수를 모형에 적합 해보았다. 이 경우 불순도 측도로는 지니계수를, 임의 추출될 설명 변수 개수 q의 최대치는 10으로 설정되었다. F1-값은 각 micro와 macro에 대하여 0.83와 0.46, PR-AUC값은 0.92, ROC-AUC값은 0.5로, 앞선 로짓 모형이나 SVM의 결과와 유사하나, micro F1-값의 경우 미세하게 하락하였다. 실제로 오차행렬을 분석한 결과, 거의 모든 데이터에 대해 '긍정'이라고 분류했으며, 실제로 '부정'인 총 486건 중 단 두 건만 제대로 분류하였음을 확인했다. 그럼에도, 앞선 일반 의사 결정 나무 모형보단 성능이 좋아졌음을 확인할 수 있다. 그 결과는 아래에서 확인할 수 있다.

```
{'criterion': 'entropy', 'max_features': 'log2'}
Test:
 F1 score(micro) / F1 score(macro) / PR-AUC / ROC-AUC
 0.83486          / 0.455   / 0.91764     / 0.49959
```

<교차 검증 및 모형 성능 결과>



<오차 행렬>

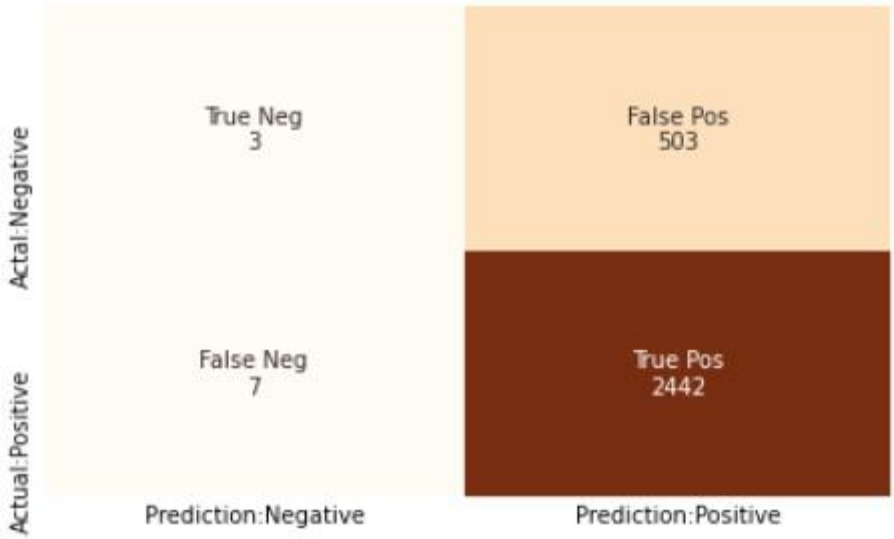


<Precision-Recall Curve과 PR-AUC>

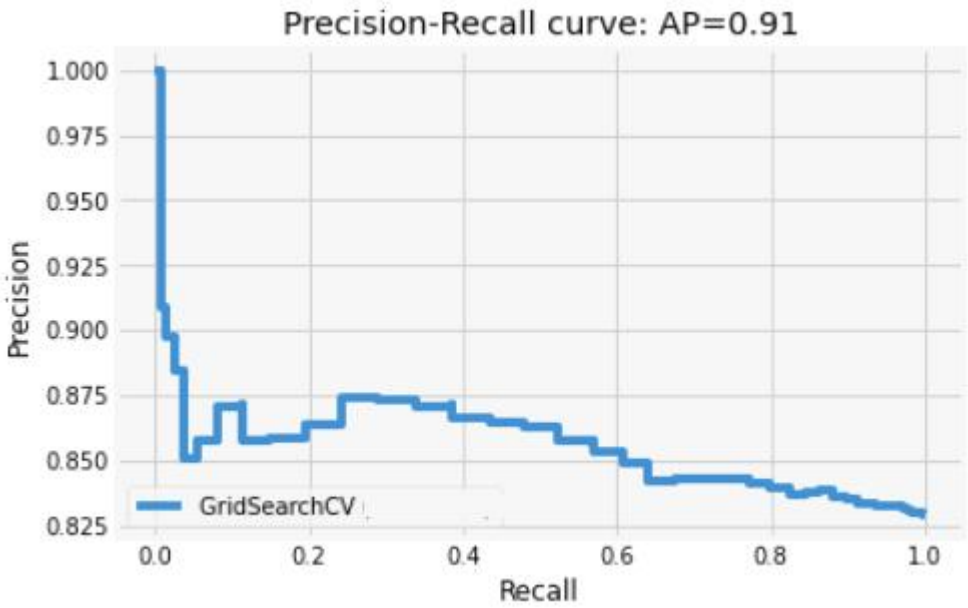
동일한 조건 아래 TF-IDF 모델로 변환한 X 변수를 적합해 보았을 때도, 앞선 의사 결정 나무 및 다른 모델들과 비슷한 결과가 나왔다. 교차 검증 결과 불순도 측도로 지니 지수를, 임의 추출될 설명 변수 개수 q의 최대치는 디폴트 값인 $\sqrt{p}$으로, 이 경우 $\sqrt{27} \simeq 5$로 설정해주었다. F1-값은 각 micro와 macro인 경우 0.83와 0.46, PR-AUC값은 0.91, ROC-AUC값은 0.5로, 이 또한 앞선 모델들의 결과와 비슷했다. 실제로 오차 행렬을 분석한 결과, 총 506개의 실제 '부정' 데이터 중 단 3개만 제대로 분류하였으며, 상대적으로 실제 '긍정'인 데이터에 대한 오분류 개수가 많았다. 그 결과는 아래와 같다.

```
{'criterion': 'gini', 'max_features': 27}
Test:
 F1 score(micro) / F1 score(macro) / PR-AUC / ROC-AUC
 0.82707          / 0.46219   / 0.91445     / 0.5029
```

<교차 검증 및 모형 성능 결과>



<오차 행렬>



<Precision-Recall Curve과 PR-AUC>

제 3장. 결론

3.1. 분석 결과 비교

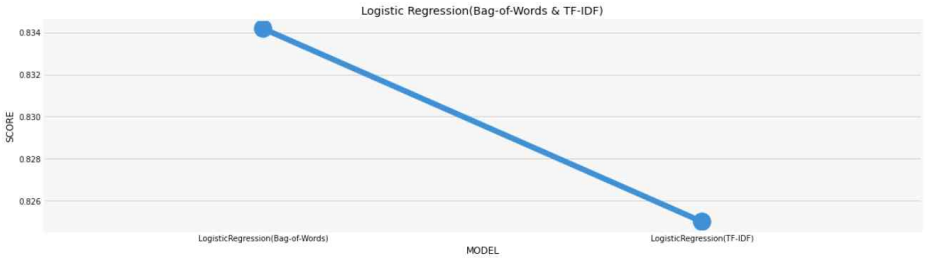
총 10번의 시뮬레이션의 결과를 평균하여, Bag-of-Words기법과 TF-IDF기법을 사용한 결과 차이 및 각 모델 성능 평가 지표를 비교해보았다.

1) F1-Score(micro)

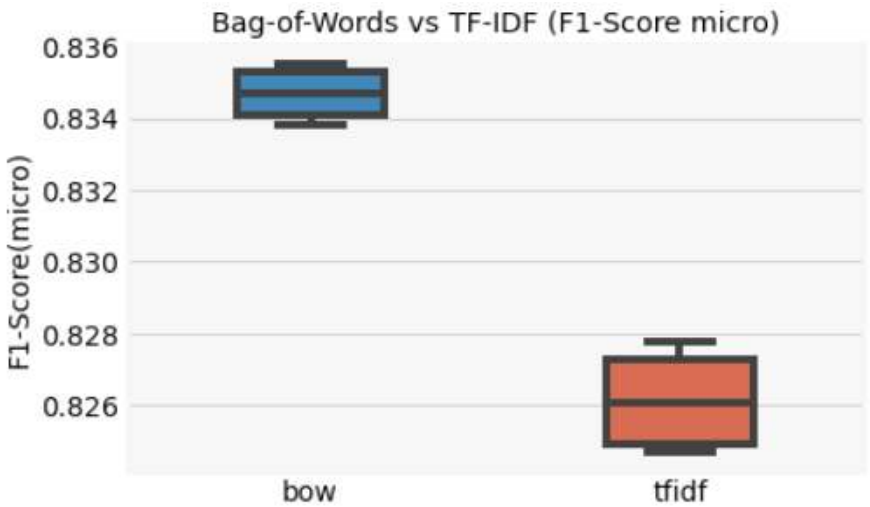
앞서 성능이 적어도 82.7% 이상이어야만 모든 경우에서 '긍정'으로 평가하는 모델보다 성능이 좋다고 할 수 있다고 말했었다. 아래 결과를 보면, TF-IDF의 성능은 이에 미치지 못하거나, 아주 유사하다. 또한, 박스-플랏(Box-Plot)으로 전체적인 분포를 비교해 보면, BoW를 사용했을 때의 F1-Score가 TF-IDF를 사용했을 때보다 모든 경우에 대해 높게 측정되었다. 따라서 본 시뮬레이션의 네 가지 모형에서 한해선 BoW 모델로 설명변수를 변환하는 것이 TF-IDF 모델보다 더 좋은 결과를 도출한다고 결론 지을 수 있었다.

Micro F1-Score의 경우, 종속변수의 각 범주의 관측치 수를 고려한 전체 성능을 평가한다. 따라서, 불균형이 존재하는 데이터에서 성능을 판단하는 지표로 흔히 쓰이며, 본 데이터의 경우 소수 클래스인 '부정'을 상대적으로 분류하지 못해도 다수 클래스인 '긍정'을 잘 분류했다면 우수하다고 판단하였다. 따라서, 앞선 시뮬레이션에서 의사 결정 나무 모델은 모든 관측치를 '긍정'으로 분류하여 실제로 '부정'인 관측치를 분류하는 데 한계가 있었지만, 그만큼 다수 클래스인 '긍정'인 관측치들에 한해선 완벽히 분류한 것이므로, 가장 성능이 좋다는 결과가 나온 것이다. 자세한 사항은 다음 장에 첨부된 그래프를 참고하길 바란다. 이를 통해 BoW 기법에 기반한 의사 결정 나무와 랜덤 포레스트 모형이 데이터 전반적으로 가장 좋은 성능을 가짐을 확인할 수 있었다.

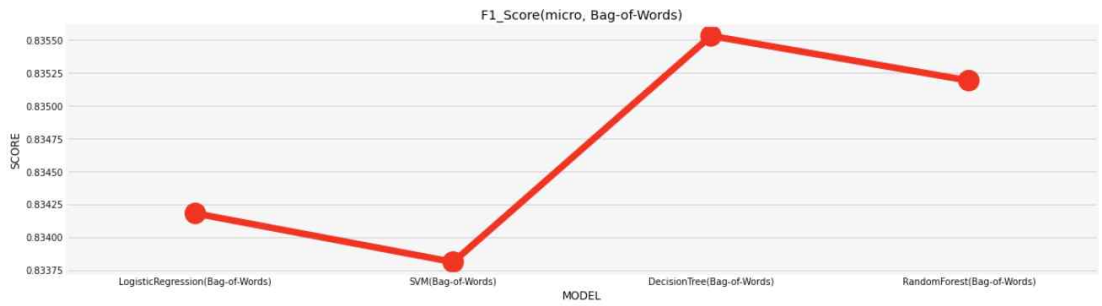
	Logistic	SVM	CART	RF
BoW	0.8342	0.8338	0.8355	0.8352
TF-IDF	0.8250	0.8247	0.8271	0.8278



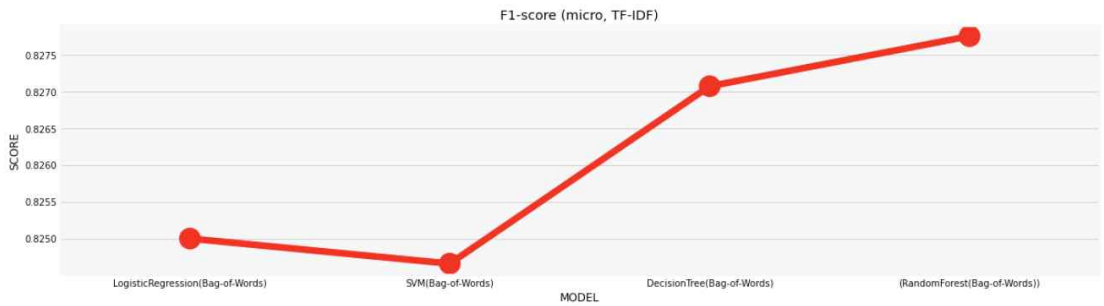
<BoW와 TF-IDF를 사용했을 때 성능 차이>



<모든 모형에 있어 BoW와 TF-IDF의 분포 비교>



<모형별 F1-Score(micro)값 비교(BoW)>



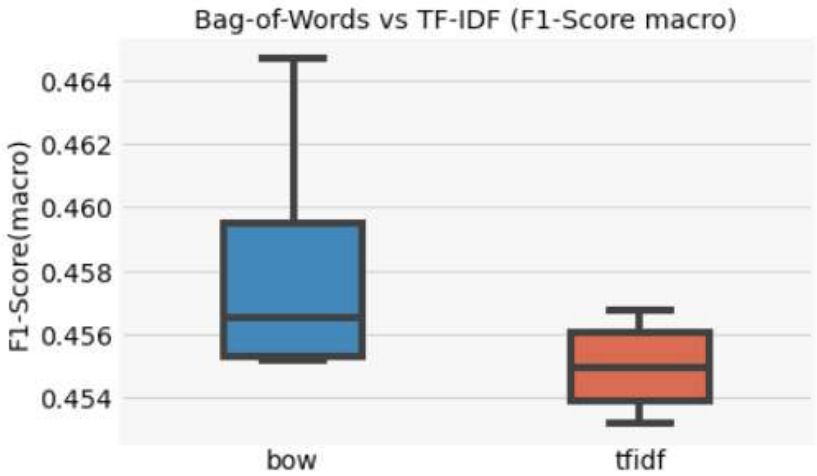
<모형별 F1-Score(macro)값 비교(TF-IDF)>

2) F1-Score(macro)

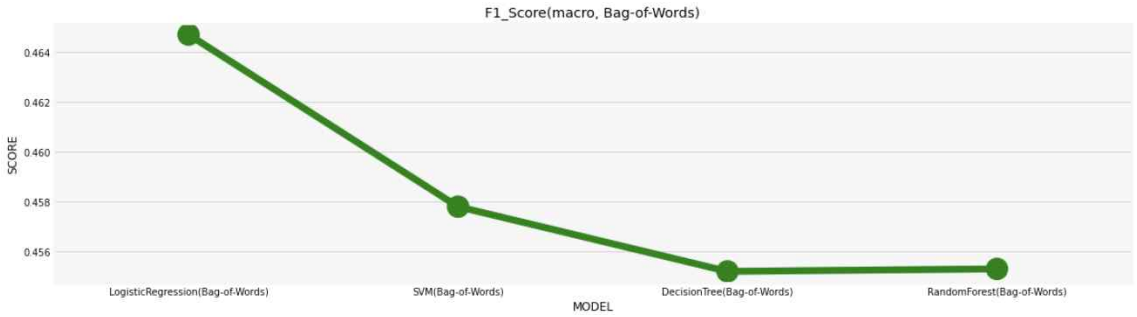
Macro F1-Score의 경우, 모든 클래스를 동등하게 취급하여, 각 클래스에서 구한 성능을 평균하기 때문에, 소수 클래스에 대한 분류 성능을 보다 중요시할 때 사용하는 지표이다. 그렇기에 앞서 좋은 평가를 받았던 의사 결정 나무 모형이 해당 지표를 사용했을 땐 가장 낮은 성능을 도출하였다. 대신, 다른 모형들에 비해 소수 클래스인 '부정'을 가장 잘 분류한 로지스틱 회귀모형이 상위권을 차지하였다. BoW과 TF-IDF 결과의 차이는 앞선 경우처럼 크진 않으나, 이 경우 역시 Bag-of-Words 기법을 사용했을 때의 성능이 전반적으로 높게 측정되었음을 확인할 수 있다. 랜덤 포레스트의 경우, 다른 모델들과 달리 TF-IDF 기법을 사용했을 때 성능이 눈에 띄게 상승했다. 자세한 사항은 아래 첨부된 그래프를 참고하길 바란다. 이를 통해 BoW 기법을 기반으로 한 로지스틱 회귀 모형과 TF-IDF 기법을 기반으로 한 랜덤 포레스트 모형이 소수 클래스를 포함한 모든 클래스에서 고루 좋은 성능을 가짐을 확인할 수 있었다.

	Logistic	SVM	CART	RF
BoW	0.4647	0.4578	04552	04553
TF-IDF	0.4558	0.4541	04532	04567

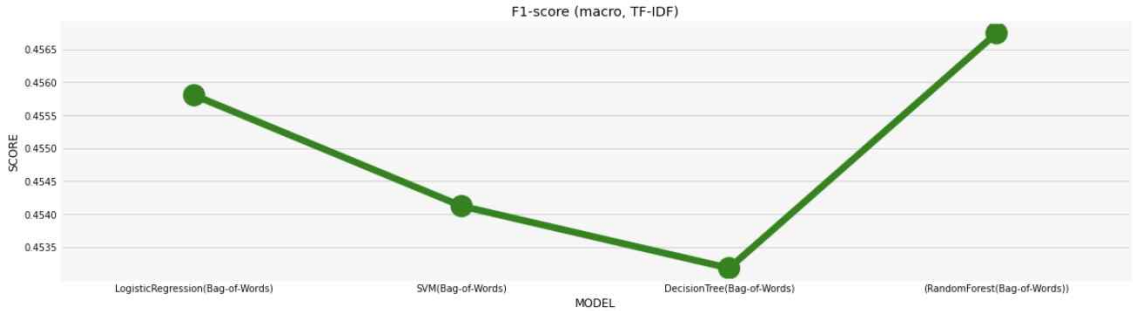
<모델별 성능 비교>



<모든 모형에 있어 BoW와 TF-IDF의 분포 비교>



<모형별 F1-Score(macro)값 비교(BoW)>



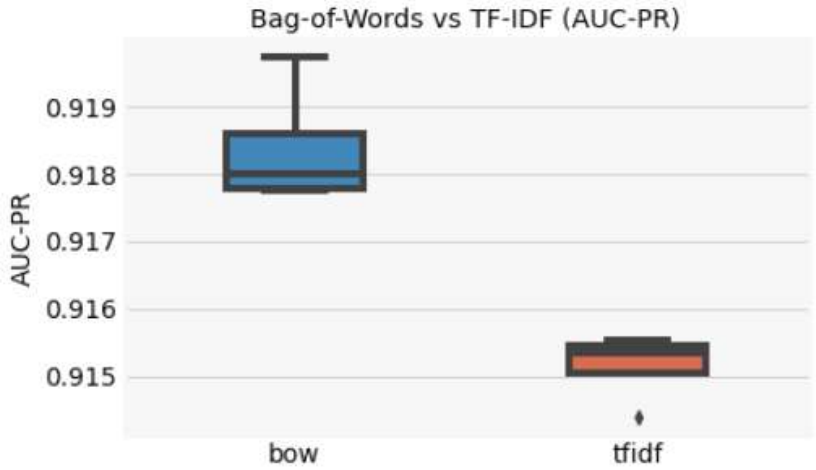
<모형별 F1-Score(macro)값 비교(TF-IDF)>

3) PR-AUC

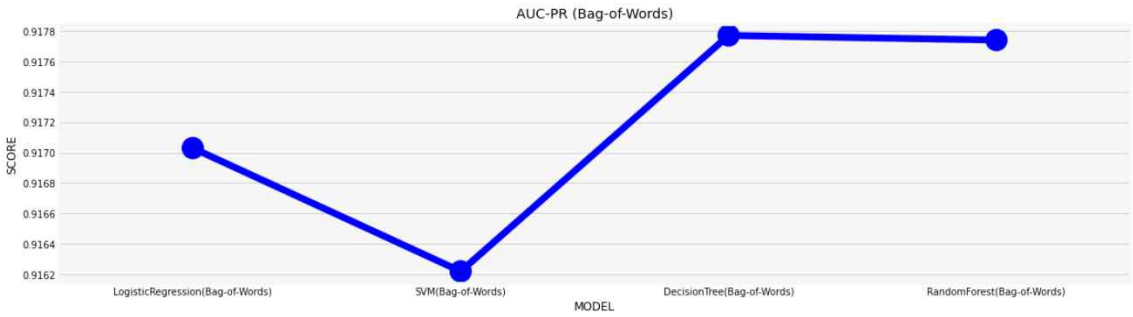
PR_AUC는 Precision과 Recall curve에 대한 면적을 나타낸 수치다. ROC_AUC가 좀 더 robust 하다고 알려졌지만, 데이터 불균형이 심한 경우, 보편적으로 PR_AUC를 사용한다. 이는 앞서 Micro F1-값처럼, 다수 클래스(+)의 올바른 분류를 우선시하기 때문이다. 본 데이터의 경우, PR-AUC는 실제 '긍정'인 데이터를 '긍정'으로 얼마나 잘 분류했는지 나타낸다. BoW과 TF-IDF 결과는 Micro F1-값의 결과와 비슷하게 다소 차이가 났다. 즉, 이 경우 역시 Bag-of-Words 기법을 사용했을 때의 성능이 전반적으로 높게 측정되었음을 확인할 수 있었다. 또한, 전반적인 모형 성능 순위도 Micro F1-값과 비슷했다. 자세한 사항은 다음 장에 첨부된 그래프를 참고하길 바란다. 이를 통해 BoW기법에 기반한 의사 결정 나무와 랜덤 포레스트 모형이 데이터 전반적으로 가장 좋은 성능을 가짐을 확인할 수 있었다.

	Logistic	SVM	CART	RF
BoW	0.9197	0.9162	0.9178	0.9177
TF-IDF	0.9138	0.9135	0.9144	0.9145

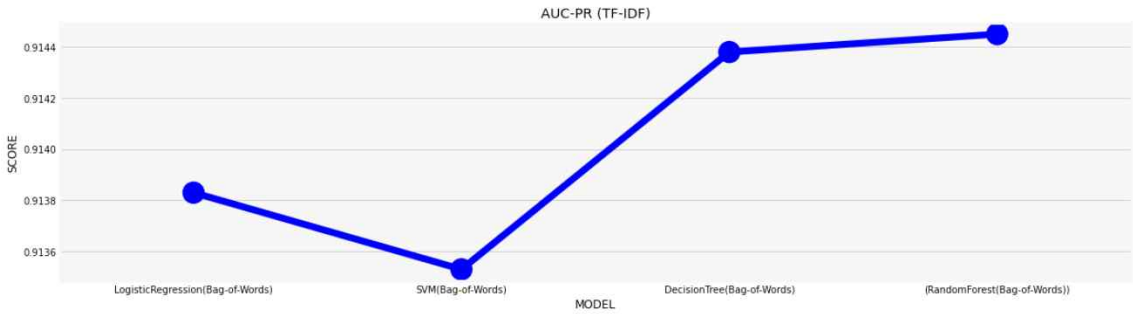
<모델별 성능 비교>



<모든 모형에 있어 BoW와 TF-IDF의 분포 비교>



<모형별 PR-AUC 값 비교(BoW)>



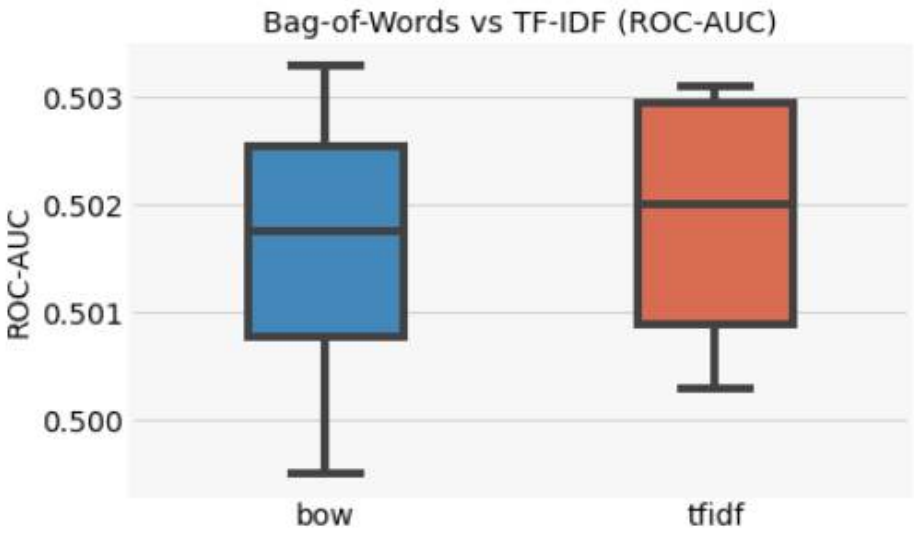
<모형별 PR-AUC 값 비교(TF-IDF)>

4) ROC-AUC

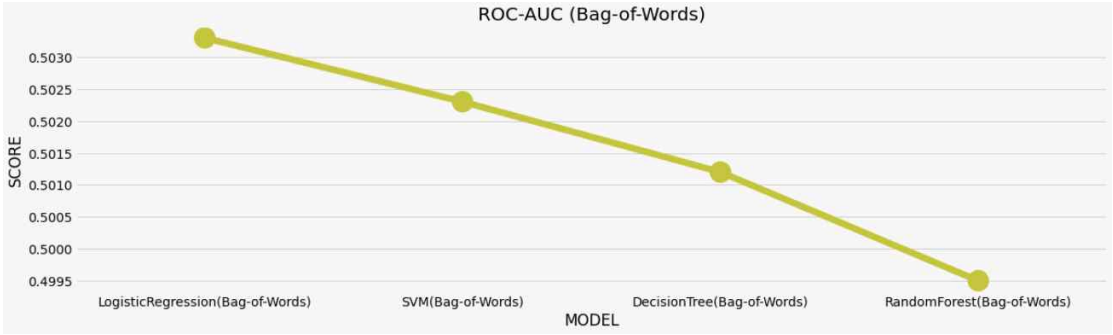
ROC-AUC는 PR-AUC와 달리, TPR(True Positive Rate)과 FPR(False Postive Rate) Curve의 면적을 나타낸다. 이 경우 앞선 Macro F1-값과 유사하게, 각 범주의 관측치 수와 관계없이 모든 범주에서 두루 성능이 좋은 모델을 찾고자 한다. 그렇기에 실제로 '부정'인 데이터를 전혀 분류하지 못했던 의사 결정 나무모형이 해당 지표를 사용했을 땐 가장 낮은 성능을 가졌다. 대신, 다른 모형들에 비해 이를 가장 잘 분류한 로지스틱 회귀모형이 상위권을 차지하였다. 다른 지표들에 비해 BoW과 TF-IDF 결과의 차이는 크진 않았다. 특히 랜덤 포레스트의 경우, BoW 기법 보다 TF-IDF 기법을 사용했을 때 성능이 눈에 띄게 상승했다. 자세한 사항은 아래를 참고하길 바란다. 이를 통해 BoW 기법을 기반으로 한 로지스틱 회귀 모형과 TF-IDF 기법을 기반으로 한 랜덤 포레스트 모형이 소수 클래스를 포함한 모든 클래스에서 고루 좋은 성능을 가짐을 확인할 수 있었다.

	Logistic	SVM	CART	RF
BoW	0.5033	0.5023	0.5012	0.4995
TF-IDF	0.5031	0.5011	0.5003	0.5029

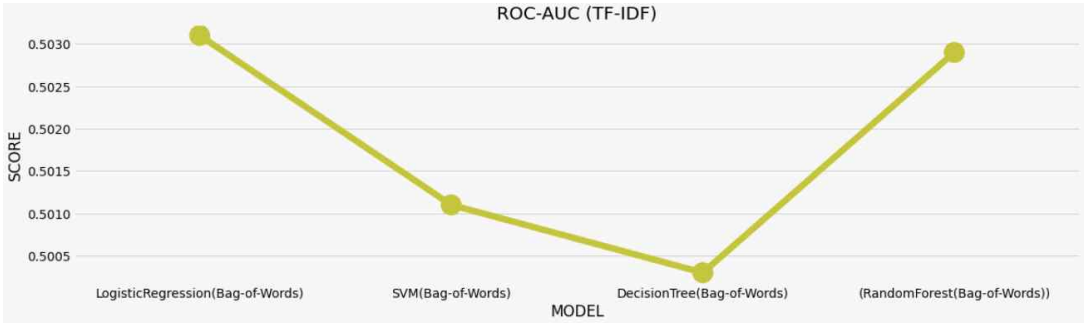
<모델별 성능 비교>



<모든 모형에 있어 BoW와 TF-IDF의 분포 비교>



<모형별 ROC-AUC 값 비교(BoW)>



<모형별 ROC-AUC 값 비교(TF-IDF)>

전반적인 위 결과를 종합하면, 아마존 리뷰 데이터로 감성 분석을 하기 위해선 TF-IDF 기법을 기반으로 한 랜덤 포레스트 모형을 사용하는 것이 가장 적합하다고 판단하였다. 로지스틱 회귀과 의사 결정 나무 모형 또한 우수하지만, 네 가지 모형 평가 척도의 상대적인 값으로 비교하였을 땐, 본 시뮬레이션에 한해선 랜덤 포레스트 모형이 최적이기 때문이다. 물론 F1-값(micro)와 PR-AUC값의 경우, BoW 기법에 기반했을 때 랜덤 포레스트 모형이 더 좋은 성능을 보이지만, 큰 차이가 나지 않았다.

이와 더불어 연구 목적에 따라 최적의 모형을 선택할 수도 있다. 앞서 언급했듯, 만약 데이터의 대다수인 '긍정'인 리뷰를 분류하는 것을 우선 목표로 둔다면, BoW 기법에 기반한 의사결정 나무 모형을, 리뷰 중 극소수인 '부정'인 리뷰를 분류하는 것을 우선으로 한다면 BoW 기법에 기반한 로지스틱 회귀모형을 선택할 수 있을 것이다.

3.2. 향후 연구 방향

본 연구에서는 아마존 Fine Foods 리뷰 데이터를 이용하여 감성 분석에 사용되는 여러 기계학습 모형에 관한 연구를 진행하였다. 이제 프로젝트를 진행하며 느낀 한계점과 앞으로 나아갈 연구 방향을 논하겠다.

일반적으로 감성 분석은 대상 텍스트 데이터에 대한 감정을 '긍정'과 '부정'으로 이진 분류한다. 본 프로젝트 또한 마찬가지인데, 현실에선 특정 문서나 문단을 하나의 감성으로 요약하기는 쉽지도, 바람직하지도 않다. 따라서, 감성 분석을 보다 효율적으로 응용하기 위해 향후 다양한 감성을 분석할 수 있는 다범주 감성 분석 방법론을 연구할 계획이다.

그러기 위해선, 단지 기계학습 모델뿐만 아니라. 감성 분석에 사용되는 딥러닝 모델을 추가로 연구해야 할 것이다. 특히, 최근엔 합성곱신경망(Convolutional Neural networks, CNN)과 순환신경망(Recurrent Neural Networks, RNN) 등의 딥러닝 모델을 감성 분석에 적용하여 기존 방법론의 성능을 향상하는 기법이 활발히 연구되고 있다. 이러한 추세에 따라, 향후 딥러닝과 접목된 다범주 감성 분석을 연구하고 싶다.

부록

1. 참고문헌

김유신, 김남규, 정승렬, *뉴스와 주가: 빅데이터 감성분석을 통한 지능형 투자의사결정모형,* 한국지능정보시스템학회

김한민, 박경보. *앙상블 기법을 활용한 온라인 음식 상품 리뷰 감성 분석*, Journal of Digital Convergence

이득환, 강형구, 이창민. *빅데이터(Big-data)에 나타난 감성 분석*, 한국금융공학회 학술발표논문집

이종화, 레환수, 이현규, *오피니언 마이닝을 통한 국내와 수입 의류 제품에 대한 고객 평판 연구*, 인터넷전자상거래연구회

이재원·박미라·유한나 공저, 『생명과학연구를 위한 통계적 방법』, 자유아카데미

허명회, 『응용데이터분석』, 자유아카데미

홍태호, 이태원, 리징징, *온라인 주식 포럼의 핫토픽 탐지를 위한 감성분석 모형의 개발*, 한국지능정보시스템학회

홍태호, 김은미, 차은정, 뉴스 감성분석과 SVM을 이용한 다우존스 지수와 S&P500 지수 예측, 한국인터넷전자상거래학회

Costello, Francis Joseph; Kun Chang Lee, *Exploring the Sentiment Analysis of Electric Vehicles Social Media Data by Using Feature Selection Methods*, Journal of Digital Convergence. 2020,

Islam, Ahsan, and Wang, *Dimensionality Reduction for Sentiment Classification using Machine Learning Classifiers*, 2019 IEEE Symposium Series on Computational Intelligence (SSCI)

Chou, *Compute the AUC of Precision-Recall Curve*,
https://sinyi-chou.github.io/python-sklearn-precision-recall/

Das, *Social Media Sentiment Analysis using Machine Learning*,
https://towardsdatascience.com/social-media-sentiment-analysis-part-ii-bcacca5aaa39

Ismiguzel,NLP-with-Python/Text-Classification.
https://github.com/Idilismiguzel/NLP-with-Python/blob/master/Text-Classification

Ismiguzel, Applying Text Classification Using Logistic Regression,
https://medium.com/analytics-vidhya/applying-text-classification-using-logistic-regression-a-comparison-between-bow-and-tf-idf-1f1ed1b83640

https://www.kaggle.com/general/7517

2. 코드

```
# EDA
import matplotlib.pyplot as plt
import nltk
import numpy as np
import pandas as pd
import seaborn as sns
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib notebook
%matplotlib inline
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction import text
import pickle
import warnings
warnings.filterwarnings("ignore")
import string
from tqdm import tqdm
from bs4 import BeautifulSoup
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import Normalizer
from sklearn.model_selection import GridSearchCV,train_test_split
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc, average_precision_score,
precision_recall_curve, plot_precision_recall_curve, f1_score
from wordcloud import WordCloud
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
```

```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import statsmodels.api as sm
from scipy import stats
from sklearn.decomposition import PCA

df = pd.read_csv('Data.csv', index_col=0)
df = df.dropna()

df.shape
df.head(6)

ax = df['Score'].value_counts().plot(kind='bar', figsize=(6,6))
fig = ax.get_figure()
ax.set_title("Review Score Counts")
ax.set_xlabel('Score')
ax.set_ylabel('Counts')

labels = 'Positive', 'Negative'
sizes = [len(np.where(df['Score']>3)[0]), len(np.where(df['Score']<3)[0])]
fig, ax = plt.subplots()
fig.patch.set_facecolor('white')
ax.pie(sizes, labels=labels, autopct='%1.1f%%',shadow=True, startangle=90)
ax.axis('equal')
ax.set_title('Percentage of Positive and Negative Reviews')
plt.show()

data =df.sample(n=10000, random_state=920)
data.shape

ax = data['Score'].value_counts().plot(kind='bar', figsize=(6,6))
fig = ax.get_figure()
```

```
ax.set_title("Review Score Counts")
ax.set_xlabel('Score')
ax.set_ylabel('Counts')

labels = 'Positive', 'Negative'
sizes = [len(np.where(data['Score']>3)[0]), len(np.where(data['Score']<3)[0])]
fig, ax = plt.subplots()
fig.patch.set_facecolor('white')
ax.pie(sizes, labels=labels, autopct='%1.1f%%',shadow=True, startangle=90)
ax.axis('equal')
ax.set_title('Percentage of Positive and Negative Reviews')
plt.show()

# Data cleansing, preprocessing
data.drop_duplicates(subset={'UserId', 'ProfileName', 'Time', 'Text'}, inplace=True)
data.shape

data.drop(['ProductId', 'UserId', 'ProfileName',
'HelpfulnessNumerator','HelpfulnessDenominator', 'Time'], axis = 1, inplace=True)
def decontracted(phrase):
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase

stopwords = set(stopwords.words('english'))
```

```
stopwords.update(["br","href"])

txt = []
for sent in tqdm(data['Text'].values):
    sent = re.sub(r"http\S+", "", sent)
    sent = BeautifulSoup(sent, 'lxml').get_text()
    sent = decontracted(sent)
    sent = re.sub("\S*\d\S*", "", sent).strip()
    sent = re.sub('[^A-Za-z]+', ' ', sent)
    sent = " ".join(e.lower() for e in sent.split() if e.lower() not in stopwords)
    txt.append(sent.strip())

sum = []
for sent in tqdm(data['Summary'].values):
    sent = re.sub(r"http\S+", "", sent)
    sent = BeautifulSoup(sent, 'lxml').get_text()
    sent = decontracted(sent)
    sent = re.sub("\S*\d\S*", "", sent).strip()
    sent = re.sub('[^A-Za-z]+', ' ', sent)
    sent = " ".join(e.lower() for e in sent.split() if e.lower() not in stopwords)
    sum.append(sent.strip())

data['Text'] = txt
data['Summary'] = sum

data = data.reset_index(inplace=False)

data.drop(['Id'], axis=1, inplace=True)
data.head()

txt = data.Summary[np.where(data['Score']>3)[0]].str.cat()
wordcloud = WordCloud(background_color='white').generate(txt)
plt.figure(figsize=(10,10))
```

```
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Summary Reviews with Positive Scores',fontsize=20)
plt.show()

txt = data.Text[np.where(data['Score']>3)[0]].str.cat()
wordcloud = WordCloud(background_color='white').generate(txt)
plt.figure(figsize=(10,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title('Text Reviews with Positive Scores',fontsize=20)
plt.show()

txt = data.Summary[np.where(data['Score']<3)[0]].str.cat()
stopwords.update(["good","great","like"])
wordcloud = WordCloud(stopwords=stopwords).generate(txt)
plt.figure(figsize=(10,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Summary Reviews with Negative Scores',fontsize=20)
plt.axis("off")
plt.show()

txt = data.Text[np.where(data['Score']<3)[0]].str.cat()
wordcloud = WordCloud(stopwords=stopwords).generate(txt)
plt.figure(figsize=(10,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Text Reviews with Negative Scores',fontsize=20)
plt.axis("off")
plt.show()

# VADER
plt.style.use('fivethirtyeight')
cp = sns.color_palette()
```

```
analyzer = SentimentIntensityAnalyzer()
sent=[]
for row in data['Text']:
    vs=analyzer.polarity_scores(row)
    sent.append(vs)
sent=pd.DataFrame(sent)
sent.head()

data = pd.concat([data.reset_index(drop=True), sent['compound']], axis=1)
data['Sentiment'] = np.where(data['compound'] >= 0 , 1, 0)

data.Sentiment[np.where(data['Score']<3)[0]]=0

len(np.where(data['Sentiment']==0)[0])
len(np.where(data['Score']<3)[0])



labels = 'Positive', 'Negative'
sizes = [len(np.where(data['Sentiment']==1)[0]),
len(np.where(data['Sentiment']==0)[0])]
fig, ax = plt.subplots()
fig.patch.set_facecolor('white')
ax.pie(sizes, labels=labels, autopct='%1.1f%%',shadow=True, startangle=90)
ax.axis('equal')
ax.set_title('Percentage of Positive and Negative Reviews')
plt.show()

data.head()

# Bag of Words
bow_converter = CountVectorizer(tokenizer=lambda doc: doc)
x = bow_converter.fit_transform(data['Text'])
words = bow_converter.get_feature_names()
```

```
len(words)

bigram_converter = CountVectorizer(tokenizer=lambda doc: doc, ngram_range=[2,2], lowercase=False)
x2 = bigram_converter.fit_transform(data['Text'])
bigrams = bigram_converter.get_feature_names()
len(bigrams)

trigram_converter = CountVectorizer(tokenizer=lambda doc: doc, ngram_range=[3,3], lowercase=False)
x3 = trigram_converter.fit_transform(data['Text'])
trigrams = trigram_converter.get_feature_names()
len(trigrams)

sns.set_style("white")
counts = [len(words), len(bigrams), len(trigrams)]
plt.plot(counts, color='blue')
plt.plot(counts, 'bo')
plt.ticklabel_format(style = 'plain')
plt.xticks(range(3), ['unigram', 'bigram', 'trigram'])
plt.tick_params(labelsize=14)
plt.title('Number of ngrams', {'fontsize':16})
plt.show()

df_bow = pd.DataFrame(x.todense())
df_bow.head()

# TF-DIF
tfidf = text.TfidfTransformer(norm=None)
X_tfidf = tfidf.fit_transform(x)
df_tfidf = pd.DataFrame(X_tfidf.todense())
df_tfidf.head()
```

```
x_train_bow, x_test_bow, y_train_bow, y_test_bow = train_test_split(df_bow, data['Sentiment'],test_size=0.3,random_state=9)
x_train_tfidf, x_test_tfidf, y_train_tfidf, y_test_tfidf = train_test_split(df_tfidf, data['Sentiment'],test_size=0.3,random_state=20)

# Logistic Regression
def logistic_cv(X_train, X_test, y_train, y_test):
    param_grid_logistic={'C':[1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3]}
    pca = PCA(n_components=10)
    X_train = pca.fit_transform(X_train)
    X_test = pca.transform(X_test)
    logistic_cv=GridSearchCV(LogisticRegression(penalty = 'l2'), param_grid_logistic, cv=5)
    logistic_cv.fit(X_train,y_train)
    model = sm.Logit(y_train, X_train)
    mod = model.fit()
    print(mod.summary())
    print(logistic_cv.best_params_)
    logistic_test=logistic_cv.predict(X_test)
    precision, recall, thresholds = precision_recall_curve(y_test, logistic_test)
    auc_precision_recall = auc(recall, precision)
    print("Test:\n","F1 score(micro) / F1 score(macro) / PR-AUC \n",round(f1_score(y_test,logistic_test, average="micro"),2),' /' ,
          round(f1_score(y_test,logistic_test, average="macro"),2),' /' ,
          round(auc_precision_recall,2))
    test_confusion_matrix=confusion_matrix(y_test, logistic_test)
    group_names = ['True Neg','False Pos','False Neg','True Pos']
    group_counts = ['{0:0.0f}'.format(value) for value in
                    test_confusion_matrix.flatten()]
    labels = [f'{v1}\n{v2}' for v1, v2 in
              zip(group_names,group_counts)]
    labels = np.asarray(labels).reshape(2,2)
    print("Test confusion matrix")
```

```
    sns.heatmap(test_confusion_matrix, annot=labels, fmt='',
cmap='Oranges',cbar=False, xticklabels=['Prediction:Negative', 'Prediction:Positive'],
yticklabels=['Actal:Negative', 'Actual:Positive'])
    disp = plot_precision_recall_curve(logistic_cv, X_test, y_test)
    disp.ax_.set_title('Precision-Recall curve: '
                       'AP={0:0.2f}'.format(auc_precision_recall))

logistic_cv(x_train_bow, x_test_bow, y_train_bow, y_test_bow)
logistic_cv(x_train_tfidf, x_test_tfidf, y_train_tfidf, y_test_tfidf)

# SVM
def svm_cv(X_train, X_test, y_train, y_test):
    pca = PCA(n_components=10)
    X_train = pca.fit_transform(X_train)
    X_test = pca.transform(X_test)
    param_grid_svm={'C':[10**(i+1) for i in range(-2,2)], 'gamma':[10**(i+1) for i in
range(-3,2)],'kernel':['rbf','sigmoid']}
    svm_cv=GridSearchCV(SVC(random_state=1206), param_grid_svm, cv=5)
    svm_cv.fit(X_train,y_train)
    print(svm_cv.best_params_)
    svm_test=svm_cv.predict(X_test)
    precision, recall, thresholds = precision_recall_curve(y_test, svm_test)
    auc_precision_recall = auc(recall, precision)
    print("Test:\n","F1 score(micro) / F1 score(macro) / PR-AUC
\n",round(f1_score(y_test,svm_test, average="micro"),2),'  /' ,
          round(f1_score(y_test,svm_test, average="macro"),2),'  /' ,
          round(auc_precision_recall,2))
    test_confusion_matrix=confusion_matrix(y_test, svm_test)
    group_names = ['True Neg','False Pos','False Neg','True Pos']
    group_counts = ['{0:0.0f}'.format(value) for value in
                    test_confusion_matrix.flatten()]
    labels = [f'{v1}\n{v2}' for v1, v2 in
              zip(group_names,group_counts)]
```

```
    labels = np.asarray(labels).reshape(2,2)
    print("Test confusion matrix")
    sns.heatmap(test_confusion_matrix, annot=labels, fmt='',
cmap='Oranges',cbar=False, xticklabels=['Prediction:Negative', 'Prediction:Positive'],
yticklabels=['Actal:Negative', 'Actual:Positive'])
    disp = plot_precision_recall_curve(svm_cv, X_test, y_test)
    disp.ax_.set_title('Precision-Recall curve: '
                       'AP={0:0.2f}'.format(auc_precision_recall))

svm_cv(x_train_bow, x_test_bow, y_train_bow, y_test_bow)
svm_cv(x_train_tfidf, x_test_tfidf, y_train_tfidf, y_test_tfidf)

# CART
def cart_cv(X_train, X_test, y_train, y_test):
    pca = PCA(n_components=10)
    X_train = pca.fit_transform(X_train)
    X_test = pca.transform(X_test)
    param_grid_cart={'criterion': ['gini', 'entropy'], 'max_depth': [2,4,6,8,10,12]}
    cart_cv=GridSearchCV(DecisionTreeClassifier(random_state=1), param_grid_cart,
cv=5)
    cart_cv.fit(X_train,y_train)
    print(cart_cv.best_params_)
    cart_test=cart_cv.predict(X_test)
    precision, recall, thresholds = precision_recall_curve(y_test, cart_test)
    auc_precision_recall = auc(recall, precision)
    print("Test:\n","F1 score(micro) / F1 score(macro) / PR-AUC
\n",round(f1_score(y_test,cart_test, average="micro"),2),'  /' ,
          round(f1_score(y_test,cart_test, average="macro"),2),'  /' ,
          round(auc_precision_recall,2))
    test_confusion_matrix=confusion_matrix(y_test, cart_test)
    group_names = ['True Neg','False Pos','False Neg','True Pos']
    group_counts = ['{0:0.0f}'.format(value) for value in
                    test_confusion_matrix.flatten()]
```

```
    labels = [f'{v1}\n{v2}' for v1, v2 in
                zip(group_names,group_counts)]
    labels = np.asarray(labels).reshape(2,2)
    print("Test confusion matrix")
    sns.heatmap(test_confusion_matrix, annot=labels, fmt='',
cmap='Oranges',cbar=False, xticklabels=['Prediction:Negative', 'Prediction:Positive'],
yticklabels=['Actal:Negative', 'Actual:Positive'])
    disp = plot_precision_recall_curve(cart_cv, X_test, y_test)
    disp.ax_.set_title('Precision-Recall curve: '
                       'AP={0:0.2f}'.format(auc_precision_recall))

cart_cv(x_train_bow, x_test_bow, y_train_bow, y_test_bow)
cart_cv(x_train_tfidf, x_test_tfidf, y_train_tfidf, y_test_tfidf)

# Random Forest
def rf_cv(X_train, X_test, y_train, y_test):
    pca = PCA(n_components=10)
    X_train = pca.fit_transform(X_train)
    X_test = pca.transform(X_test)
    param_grid_rf={'criterion':["gini","entropy"]
,'max_features':['sqrt','log2',X_train.shape[1]]}
    rf_cv=GridSearchCV(RandomForestClassifier(random_state=100), param_grid_rf,
cv=5)
    rf_cv.fit(X_train,y_train)
    print(rf_cv.best_params_)
    rf_test=rf_cv.predict(X_test)
    precision, recall, thresholds = precision_recall_curve(y_test, rf_test)
    auc_precision_recall = auc(recall, precision)
    print("Test:\n","F1 score(micro) / F1 score(macro) / PR-AUC
\n",round(f1_score(y_test,rf_test, average="micro"),2),'  /' ,
          round(f1_score(y_test,rf_test, average="macro"),2),'  /' ,
          round(auc_precision_recall,2))
    test_confusion_matrix=confusion_matrix(y_test, rf_test)
```

```
    group_names = ['True Neg','False Pos','False Neg','True Pos']
    group_counts = ['{0:0.0f}'.format(value) for value in
                    test_confusion_matrix.flatten()]
    labels = [f'{v1}\n{v2}' for v1, v2 in
              zip(group_names,group_counts)]
    labels = np.asarray(labels).reshape(2,2)
    print("Test confusion matrix")
    sns.heatmap(test_confusion_matrix, annot=labels, fmt='',
cmap='Oranges',cbar=False, xticklabels=['Prediction:Negative', 'Prediction:Positive'],
yticklabels=['Actal:Negative', 'Actual:Positive'])
    disp = plot_precision_recall_curve(rf_cv, X_test, y_test)
    disp.ax_.set_title('Precision-Recall curve: '
                       'AP={0:0.2f}'.format(auc_precision_recall))

rf_cv(x_train_bow, x_test_bow, y_train_bow, y_test_bow)
rf_cv(x_train_tfidf, x_test_tfidf, y_train_tfidf, y_test_tfidf)

#comparison graph
plt.figure(figsize=(18,5))
plt1 = sns.pointplot(x='Model',y='F1_Score(micro)',data=compare, color="r")
plt2 = sns.pointplot(x='Model',y='F1_Score(macro)',data=compare, color="g")
plt3 = sns.pointplot(x='Model',y='PR-AUC',data=compare, color="b")
plt.title('PR-AUC (Bag-of-Words)')
plt.xlabel('MODEL')
plt.ylabel('SCORE')
plt.show()

%matplotlib inline
ax = sns.boxplot(data=search_results, width=0.4)
ax.set_ylabel('F1-Score(micro)', size=14)
ax.set_title('Bag-of-Words vs TF-IDF (F1-Score micro)')
ax.tick_params(labelsize=14)
```