[Project] 중고나라 사기글 판별

1조: 18 수학과 김현주
19 산업경영공학과 박서인

[목차]

I. 주제 선정 계기

중고나라에서 사기가 기승을 부리고 있고, 그 피해가 좀처럼 줄지 않아 중고나라 회원들끼리 오픈채팅방을 만들어서 소송을 준비하는 등, 중고나라 차원에서의 소비자 보호가 필요한 시점이다. 중고나라는 2018년도부터 네이버와 협력하여 사기예방 솔루션 "레드카드"를 도입하고 있지만, 굉장히 방어적이고 정형화된 패턴으로 이를 악용해서 솔루션의 감시망을 피해가는 사기꾼들이 갈수록 증가하고 있기 때문에 새로운 솔루션으로의 대체가 시급해보인다. 사기글에서 육안으로 관찰될 수 있는 의심스러운 정황(작성자에 대한 정보 부족, 과다한 홍보, 낮은 가격 등)이 있기 때문에, 이런 feature들을 다양한 방식으로 추출하여 **"사기글을 판별하는 새로운 모델"**을 제작하는 것이 위 프로젝트의 목표이다. 좀 더 효율적이고 정확한 솔루션의 개발은 이용자들로 하여금 온라인 중고거래 사기 피해를 예방하고, 중고나라(플랫폼)으로 하여금 사기 의심글 필터링을 통한 (이용자들의) 플랫폼 만족도를 제고할 수 있을 것이다. 종래에는 중고 온라인 플랫폼에서의 안전한 거래 문화를 정립하고, 서비스 질을 향상시킬 것으로 기대된다.

II. 데이터 수집

이 프로젝트의 상당히 많은 시간을 데이터 수집에 할애했다. 데이터 수집을 할 품목 선정, feature 선정, 모을 데이터 개수, 사기글과 정상글 판별 방법에 대해 여러 가지를 고려하고 논의하였다.

1. 품목 선정(검색어): 에어팟 프로 미개봉

사기가 빈번히 일어나는 '전자기기' 내에서 선정하였으며, 여러 품목을 선정하면 각 품목의 가격 등의 feature을 동일선상에서 비교하기 어렵다는 문제가 있었기 때문에, 한 가지 품목만을 선정하였다. 품목 선정 기준은 다음과 같다.

1. ‘품목명’이 간결하며, ‘품목명’을 검색했을 때 다른 품목이 섞이지 않고 최대한 그 품목만이 검색되는 것

ex. 아이폰 11을 검색하면, 아이폰 11 mini, 아이폰 11 pro등이 같이 검색되기 때문에 부적합. 비슷한 이유로 폰 종류 제외. 에어팟 1,2세대 또한, ‘유선충전’, ‘무선충전’이 섞여서 검색되어 제외.

2. 그 품목이 다른 악세사리와 같이 판매되는 경우가 적은 것

ex. 아이패드의 경우 애플펜슬, 필름, 케이스 등과 같이 판매되는 경우가 많음. 비슷한 경우로 태블릿 제외

3. 근 1년간 동일 품목의 신상품이 출시된 적이 없어 가격의 변동폭이 작은 것

ex. 아이패드 pro의 경우 5세대가 출시되어 가격이 많이 내려감

4. 가격이 적당히 낮아서 사기글이 비교적 많이 있는 것

5. 중고가 방어가 잘 되어 가격 하락 폭이 작은 것

ex. 버즈 프로는 2달 내에 2만원 정도의 하락폭을 겪음. 비슷한 이유로 중고가 방어가 잘 안되는 갤럭시 제품 제외

위의 기준에 가장 잘 부합하고, 특히 근 1년 내 중고가 변동 폭이 1만원 이내인 에어팟 프로를 선정하였다. 가격의 통일성을 위해 ‘미개봉’ 상품만을 고려하였다.

2. feature 선정

기존 proposal에서 제시했던 ‘본문 길이’, ‘작성자 가입년도와 가입일’, ‘작성경과일’, ‘본문 이미지 개수’, ‘거래금액과 시세차이’, ‘작성 빈도수’ 중 ‘작성자 가입년도와 가입일’, ‘작성경과일’은 알 수 있는 방법이 없어 제외했다. 대신 사기글을 훑어보았을 때, 텍스트 마이닝의 필요성을 느껴 특정 단어가 포함되어있는지 그 여부를 feature로 추가하였다.

feature	설명
author_num	해당 글 작성자의 전체 작성 게시글 수
price	판매 가격
length	해당 글의 텍스트 길이(여백 포함 글자 수)
img	해당 글의 이미지 개수
attend_num	해당 글 작성자의 중고나라 카페 방문 수
deliver	해당 글에 ‘직거래’ 단어 포함 여부(포함o: 1, 포함x: 0)
talk	해당 글에 ‘톡’, ‘talk’ 단어 포함 여부(포함o: 0, 포함x: 1)
spam	사기글(1), 정상글(0)

3. 웹 크롤링

웹 크롤링은 python의 Selenium library를 이용하였다. 코드 실행 순서는 다음과 같다.

1. chrome.exe로 중고나라 카페에 접속 후 menu 중 '주변기기/악세사리' 탭 클릭
2. 검색창에 '에어팟 프로 미개봉' 검색 후 클릭
3. 좌측 상단 판매완료 상품을 포함하기 위해 default 값인 판매 중이 아닌, '게시글 전체' 클릭, 우측 상단 (1페이지에) 50개씩 모아보기 클릭
4. 게시글을 하나씩 접속하며 feature 추출. 이때 접속이 안되는 게시글이나, 게시글 제목에 '워치', '2세대'가 포함된 경우 feature 값을 -1로 지정 (전처리 용이)
5. 판매완료 된 상품은 spam 값을 0으로 지정, 판매 중인 상품은 크롤링만으로 사기글 여부를 모르기 때문에 우선 2로 지정

4-5의 과정을 50페이지(크롤링할 수 있는 거의 최대치)까지 반복하여 총 2500개의 데이터를 모았다. 이후 크롤링으로 새 게시글 30개를 추가하였다.

```
if '워치' in title_temp or '2세대' in title_temp:
    problem=0

#작성 수, 가격, 그림 개수, 텍스트 글자
target1 = driver # preserve driver element

time.sleep(1)
target = target1.find_element_by_xpath(f'//*[@id="main-area"]/div[5]/table/tbody/tr[{i}]/td[1]/div[2]/div/a')
target.send_keys(Keys.CONTROL + "\n")
driver.switch_to.window(driver.window_handles[1])

time.sleep(1)

driver.switch_to.frame("cafe_main")

#가격
price_temp=driver.find_element_by_xpath('//*[@id="app"]/div/div/div[2]/div[2]/div[1]/div[1]/div[2]/div[2]/div[1]/div[1]/div/strong').text

price_temp=''.join( x for x in price_temp if x not in ",원")


if driver.find_element_by_xpath('/html/body/div/div/div/div[2]/div[2]/div[1]/div[1]/div[2]/div[2]/div[1]/div[1]/p/em').text=="완료":
    spam_temp=0
```

[Figure 1] 크롤링 코드 일부

4. 사기글 수집 및 사기글 분류

중고나라의 '불량거래 후기' tab의 사기글 신고글을 통해 148개의 데이터를 수집했다. 보통 캡쳐본으로 신고하기 때문에 크롤링이 아닌 수작업으로 feature을 추출했으며, 이때 특히 attend_num(방문 수) 값이 null인 결측치들이 생겼다. 이렇게 크롤링과 수작업으로 모은 데이터는 최종 2678개였다.

[Figure 2] 신고글 예시

spam값이 2인 데이터는 일일이 확인하면서 확실히 사기글(댓글에 '사기'라고 달려있거나, 가격이 터무니없이 싸거나, 더치트에 사기 이력이 있는 경우)인 경우는 1, 확실히 정상글인 경우(최근까지 활동 이력이 있거나 신용이 확실한 경우)는 0으로 지정했으며, 알 수 없는 경우는 지웠다.

III. 데이터 전처리

python의 pandas library를 사용했다.

1. 데이터 정제

같은 사람이 글을 여러 개 올린 경우가 많아서 반복되는 row를 제거했다. 또한, title에 '삽니다', '삼', '매입', '버즈', '갤럭시', '소니' 등의 단어가 포함된 경우를 추가로 제거했다. 위 과정 후 총 1179개의 데이터가 남았다.

2. 결측치

결측치가 문제였는데, spam data 중 attend_num이 비어있는 경우, "author_num이 1 이상 5 이하인 데이터"의 "attend_num의 median" 값으로 채웠다. attend_num이 author_num과 관련성이 크다고 판단했고, 실제로 spam data 대부분의 author_num이 한 자릿수에 불과했기 때문이다. price가 결측치인 경우는 사기글인 data에서만 나타났기 때문에 사기글 price의 median 값으로 채웠으며, deliver, talk, length, img이 없는 글이 한 개, length와 img가 없는 글이 한 개여서 위 두 데이터는 지웠다.

최종적으로 **1177개의 데이터(사기글 204개, 정상글 973개)**를 가지고 training을 실시했다.

```
In [18]: data.isnull().sum()

Out[18]: title           0
         written_time    0
         author          1
         author_num      0
         price           6
         length          2
         img             2
         attend_num     45
         deliver         1
         talk            1
         spam            0
         dtype: int64
```

[Figure 3] 결측치 현황

```
c12 = data['author_num']>=1
c13 = data['author_num']<=5
sub1 = data[c12 & c13]
att_med = sub1['attend_num'].median()
data['attend_num'].fillna(att_med, inplace=True)

data.dropna(subset=['deliver', 'img'], inplace=True)

c14 = data['spam']==1
sub2 = data[c14]
price_med = sub1['price'].median()
data['price'].fillna(price_med, inplace=True)
```

[Figure 4] 결측치 처리

3. 정규화

python의 sklearn library의 MinMaxScaler를 사용하였다.

기본적으로 talk과 deliver은 0과 1 중 하나의 값을 가지고 있기 때문에 나머지 feature들도 0에서 1 사이 값으로 normalize 시키는 게 맞다고 판단하여 scaling을 실시하였다.

> data

	author_num	price	length	img	attend_num	deliver	talk	spam
1	1	200000	132	3	314	0	1	1
2	21	230000	81	1	979	1	1	0
3	23	230000	54	1	752	0	1	0
4	1	225000	27	2	25	0	1	1
5	152	210000	28	3	3381	1	1	0
6	87	230000	77	1	4925	1	1	0
7	21	230000	106	1	979	1	1	0
8	2	220000	95	4	640	1	1	0
9	39	225000	41	1	835	0	1	0
10	61	225000	64	1	747	1	1	0
11	17	220000	43	1	1546	0	1	1
12	44	230000	56	1	1240	1	1	0
13	19	240000	27	2	777	0	1	0
14	5	220000	87	2	91	1	1	0
15	87	230000	66	1	487	1	1	0
16	44	230000	102	1	241	1	1	0
17	1	130000	70	1	333	1	1	1
18	39	225000	28	2	2145	0	1	0

[Figure 5] scaling 전

> data2

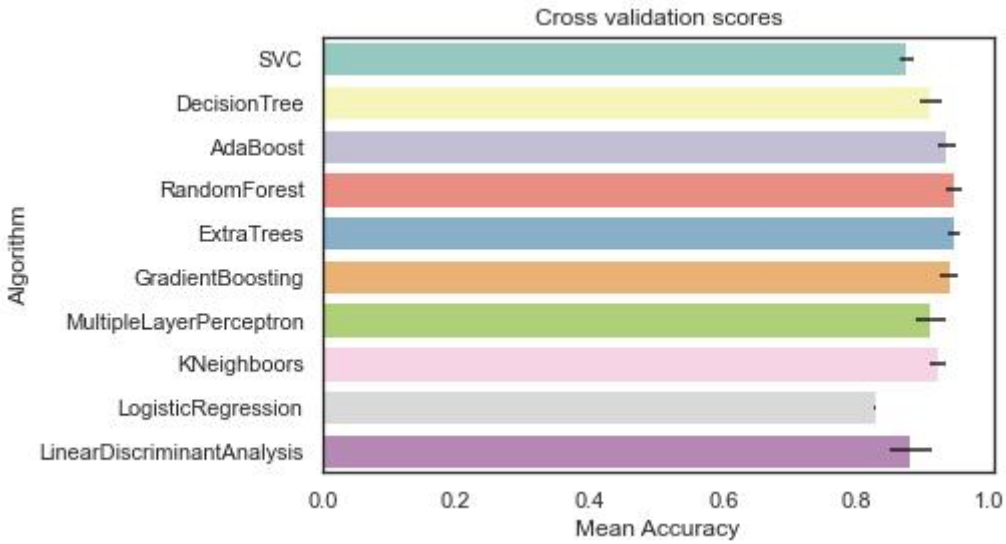
	author_num	price	length	img	attend_num	deliver	talk	spam
1	0.0000000000	0.07284768	0.12324930	0.3333333	0.0163908672	0	1	1
2	0.0027137042	0.09271523	0.07563025	0.1111111	0.0512149141	1	1	0
3	0.0029850746	0.09271523	0.05042017	0.1111111	0.0393276079	0	1	0
4	0.0000000000	0.08940397	0.02521008	0.2222222	0.0012568077	0	1	1
5	0.0204884668	0.07947020	0.02614379	0.3333333	0.1770004189	1	1	0
6	0.0116689281	0.09271523	0.07189542	0.1111111	0.2578550482	1	1	0
7	0.0027137042	0.09271523	0.09897292	0.1111111	0.0512149141	1	1	0
8	0.0001356852	0.08609272	0.08870215	0.4444444	0.0334625052	1	1	0
9	0.0051560380	0.08940397	0.03828198	0.1111111	0.0436740679	0	1	0
10	0.0081411126	0.08940397	0.05975724	0.1111111	0.0390657729	1	1	0
11	0.0021709634	0.08609272	0.04014939	0.1111111	0.0809069962	0	1	1
12	0.0058344640	0.09271523	0.05228758	0.1111111	0.0648826979	1	1	0
13	0.0024423338	0.09933775	0.02521008	0.2222222	0.0406367826	0	1	0
14	0.0005427408	0.08609272	0.08123249	0.2222222	0.0047130289	1	1	0
15	0.0116689281	0.09271523	0.06162465	0.1111111	0.0254503561	1	1	0
16	0.0058344640	0.09271523	0.09523810	0.1111111	0.0125680771	1	1	0
17	0.0000000000	0.02649007	0.06535948	0.1111111	0.0173858400	1	1	1

[Figure 6] scaling 후

IV. 모델훈련 및 결과분석

python의 sklearn library를 사용하였다. 우선 SVC(Support Vector machine Classifier), DecisionTree, AdaBoost, RandomForest, ExtraTrees, GradientBoosting, MultiLayerPerceptron, KNeighboors, LogisticRegression, LinearDiscriminantAnalysis 총 10개의 classifier을 사용하였다. 각 모델은 5-fold cross validation 방식을 이용하여 훈련되었다. train set과 test set의 비율은 8:2로, 데이터를 shuffle시켜서 랜덤하게 나누었다.

1. 정확도



[Figure 7] 10개 모델에 따른 train set의 mean accuracy

이중 정확도가 높은 상위 네 개-**RandomForest(이하 RFC), ExtraTree(이하 ExtC), AdaBoost(이하 AdaC), GradientBoosting(이하 GBC)과, SVC(이하 SVMC)**을 선택하여 다음의 과정들을 수행하였다.

우선, 다음 표는 각 모델들에 대해 best hyperparameter set을 골라서 train set과

test set에 대해 정확도를 계산한 결과이다. train set에 대한 mean accuracy는 AdaC 제외하고 94%로 거의 동일하였으며, test set의 accuracy는 모델마다 차이가 있었고 GBC와 RFC가 95%로 가장 높았다.

Classifier	train set	test set
AdaC	0.929854778790949	0.9152542372881356
ExtC	0.9479511426319936	0.9491525423728814
RFC	**0.9490149724192276**	0.9533898305084746
GBC	0.9468873128447596	**0.9576271186440678**
SVMC	0.9489980862321289	0.940677966101695

하지만 중고나라 데이터 셋의 경우 특히 정상글과 사기글의 비율이 거의 4.5:1이기 때문에 정확도 자체만으로 모델의 타당성을 평가할 수 없다. 극단적인 예로, 모든 test set의 data를 정상글로 분류하여도 정확도가 약 80%로 상당히 높게 나온다. 따라서 Confusion matrix, feature들의 Relative Importance, ROC curve와 AUC 등을 계산함으로써 모델의 타당성을 검증하고자 하였다.

2. Confusion matrix

RFC	pred(0)	pred(1)
data(0)	190	3
data(1)	8	35

ExtC	pred(0)	pred(1)
data(0)	191	2
data(1)	10	33

SVMC	pred(0)	pred(1)
data(0)	190	3
data(1)	11	32

AdaC	pred(0)	pred(1)
data(0)	188	5
data(1)	15	28

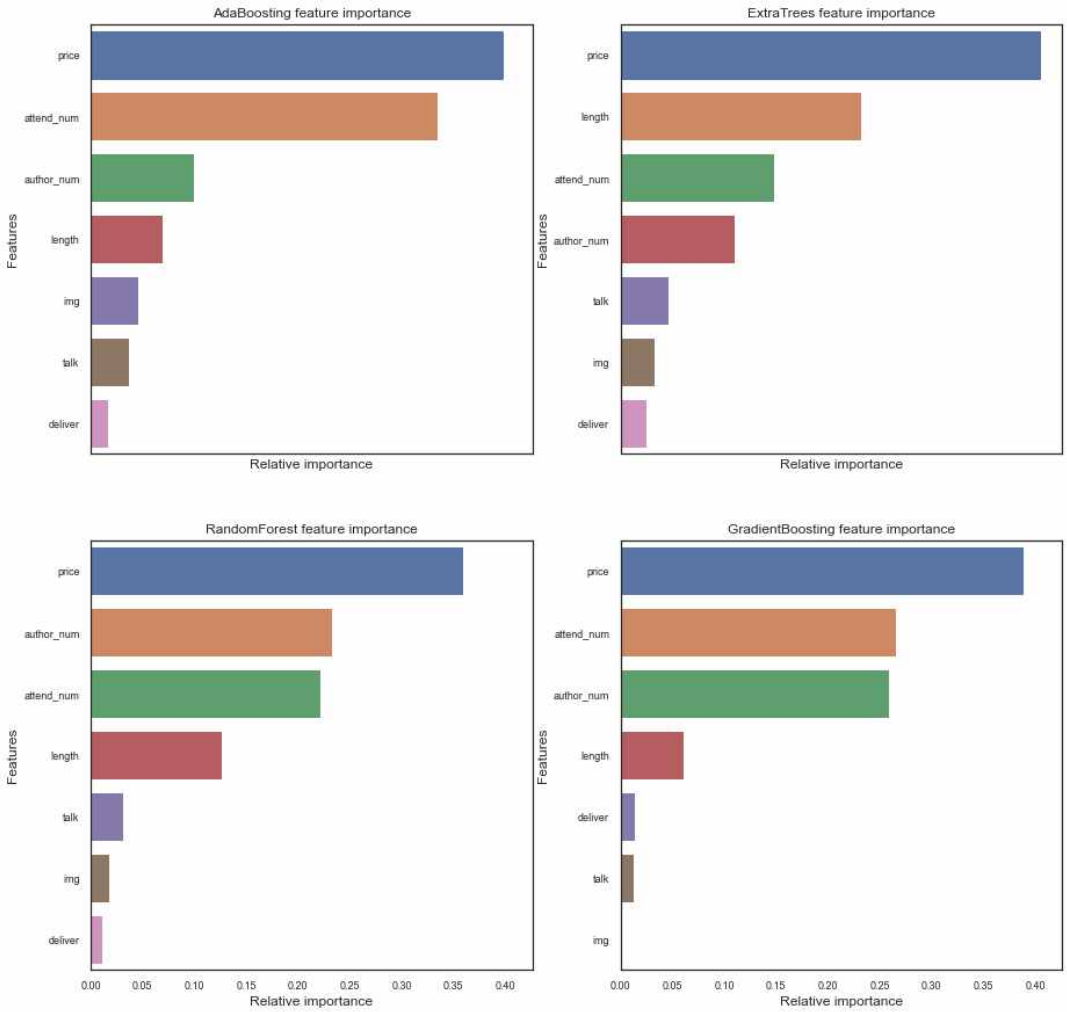
GBC	pred(0)	pred(1)
data(0)	189	4
data(1)	6	37

이중, 가장 중요한 “실제 사기글(1)이 사기글(1)로 예측될 확률(Sensitivity)(%)”은 **GBC(86%) > RFC(81%) > ExtC(77%) > SVMC(74%) > AdaC(65%)** 순으로 높았다.

3. Relative Importance

SVMC를 제외한 모델에서 Feature들의 relative importance를 구한 결과이다. 네 모델 모두 상위 네 개의 feature과 하위 3개의 feature은 같았지만, 세세한 차이를 보였다. 예를 들어 length를 네 번째로 중요한 변수로 판단한 반면, ExtC만이 length를 두 번째 중요한 변수로 뽑았다.

Classifier	1	2	3	4	5	6	7
AdaC	price	attend_num	author_num	length	img	talk	deliver
ExtC	price	length	attend_num	author_num	talk	img	deliver
RFC	price	author_num	attend_num	length	talk	img	deliver
GBC	price	attend_num	author_num	length	deliver	talk	img



[Figure 8] 4개 모델(왼쪽 위부터 시계 방향으로 AdaC, ExtC, GBC, RFC)의 Relative importance

*** 상위 네 개**

(1) price

모든 Classifier들이 가장 중요하게 뽑은 feature은 price였다. 실제로 사기글의 판매 가격은 정말 낮게는 10만원에서 22만원까지 정상글의 판매 가격에 비해 다소 낮게 형성되어 있었다. 하지만 23, 24만원으로 사기를 치는 게시글도 있었고, 급처의 명분으로 18~19만원에 판매하는 정상글도 있었기 때문에 가격만으론 사기글과 정상글이 분류되진 않는다. 이런 가격대의 게시글의 사기 여부를 파악하기 위해서 아래의 feature들이 필요하다.

(2) attend_num

보통 사기를 치는 판매자의 경우 계정을 여러 개 쓰거나 새로 파서 쓰기 때문에 "대체로" 방문수 자체는 작고, 그것을 실제 위의 Relative importance로 확인할 수 있었다.

(3) author_num

attend_num과 마찬가지로 사기글의 author_num이 대체로 작은 경향을 보였다.

(4) length

다소 신기한 결과였는데 length가 길수록 사기글로 판별될 확률이 높았다. 사기글을 훑어보았을 때, “지인에게 선물 받았는데 처분한다”, “정품 인증 가능하다”, “아이패드를 갖고 싶어서 경품을 지원했는데 이미 있는 에어팟에 당첨되어서 판매한다”, “~~ 전화번호로 문의하라”, “톡으로 문의하라” 등의 불필요한 사족들이 달려있는 경우가 많았기 때문이라고 판단했다.

*** 하위 세 개**

(5) img

img는 중요한 feature가 아니었다. 실제로 사기글과 정상글 대부분 사진의 개수가 1~5개 사이로 다양했다. 특히 미개봉 상품이라 사진을 많이 업로드할 이유가 없었던 것으로 생각된다.

(6) talk

의외의 결과였는데, 보통 카카오톡으로 사기를 치는 경우가 많아서 중요한 변수일 줄 알았지만, 사기글 중에 “톡(or talk)”이 포함되어 있지 않은 경우와, 정상글 중에 “톡으로 문의하라 그러면 대부분 사기에요” 등의 문구가 많아 talk 변수의 중요도가 희석된 것으로 보인다.

(7) deliver

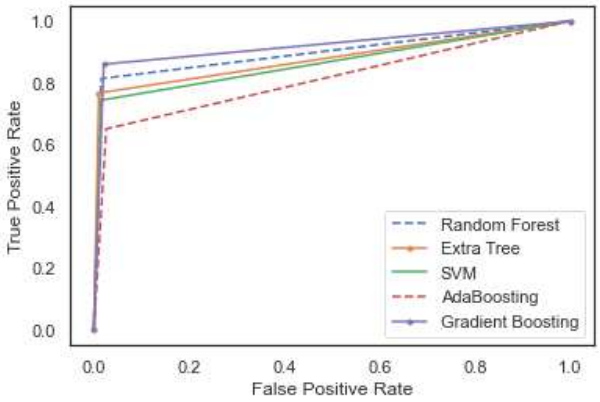
이 또한 의외의 결과였다. 하지만 사기글 중 대다수가 “직거래가 가능해요”라고 말을 하지만 실제로는 가능하지 않은 경우, “(외진 지역=전남 함평, 파주, 강원 인제 등)에서 직거래 가능해요” 등의 문구가 포함되어있는 경우가 많았기 때문에 변수의 중요도가 희석된 것으로 보인다. “직거래” 단어가 포함되어있지만 “실제로” 직거래가 가능한 경우를 단순한 text mining으로는 확인할 수 없기 때문에 생긴 문제였다.

4. ROC curve and AUC

다음은 모델의 설명력을 평가하기 위해 다섯 모델의 ROC curve와 AUC를 계산한 결과이다. 앞서 Sensitivity 값이 가장 높았던 GBC가 역시 AUC값이 가장 컸다.

Classifier	AUC(descending order)	등급
GBC	0.9198698638390168	Excellent
RFC	0.8992047234606579	Good
ExtC	0.8785395830822991	Good
SVMC	0.8643210025304253	Good
AdaC	0.8126280274731896	Good

[일반적인 기준]
0.90-1 = excellent
0.80-0.90 = good
0.70-0.80 = fair
0.60-0.70 = poor
0.50-0.60 = fail



[Figure 9] ROC curve on 5 models respectively

5. Voting

[Figure 10] heat map among 5 models

```
votingC = VotingClassifier(estimators=[('rfc', RFC_best), ('extc', ExtC_best),
('svc', SVMC_best), ('adac',ada_best),('gbc',GBC_best)], voting='hard', n_jobs=4)

votingC = votingC.fit(X_train, Y_train)
```

```
test_Spam = pd.Series(votingC.predict(X_test), name="Spam")
accuracy_score(Y_test, test_Spam)
```

```
0.9533898305084746
```

```
confusion_matrix(Y_test, test_Spam)
```

```
array([[190,   3],
       [  8,  35]], dtype=int64)
```

[Figure 11] Python code for HVC

지금까지 학습한 모델들을 이용해서 앙상블 모델을 만들고, 앞에서와 마찬가지로 여러 분석을 시행하였다. 여기서 사용한 앙상블 기법은 voting이다. 먼저 voting을 했을 때 유의미한 차이가 있을지 확인하기 위해 model간의 상관계수를 보여주는 heatmap을 plotting 했다. 대체로 0.73~0.96 사이의 유의미한 차이를 보여주었다. 또한 AdaC가 다른 Classifier간의 상관계수가 0.7x로 낮은 반면, ExtC는 다른 Classifier과의 상관계수가 비교적 높았다. 여기서는 Hard Voting Classifier(HVC)를 사용하여, 각 분류기의 예측(1-0)을 보고 다수결의 방식으로 채택하게끔 하였다. Hard Voting Classifier의 test accuracy와 Sensitivity 값은 각각 0.9533898305084746과 35/43*100=81로 기존 Classifier에 비해 높은 편에 속했다.

6. 결론

AdaBoost를 제외한 모델 모두 training accuracy와 test accuracy가 94~95%로 높았으며, AUC와 sensitivity 값 모두 준수했다. 더 많은 data set(특히 spam data)를 확보할 수 있다면 정확도가 달라지겠지만, 이 중에서 한두 가지를 실제 솔루션으로 채택을 해야한다면 test accuracy와 sensitivity 값이 가장 높았던 GBC와, 여러 모델의 장단점을 절충할 수 있는 HVC를 채택하는 것이 좋을 것이라는 최종 결론을 내렸다.

V. 의의 및 한계

1. 의의

이 프로젝트는 실제로 모델들이 높은 test accuracy와 준수한 sensitivity와 AUC 값을 가졌다는 것에서 큰 의의를 가진다. 또한 처음에 중요하다고 가정했던 각 feature들이 얼마나 중요한지를 실제로 확인했으며, 각 feature의 Relative importance 값이 높고 낮은 이유를 경험적으로 분석할 수 있었다.

2. 한계

(1) data 수집의 한계

- 기본적으로 중고나라 '이용자' 입장에서 데이터를 모았기 때문에 정보 접근에 한계가 있었다. 예를 들어, 이용자가 접근 가능한 정보는 작성자의 닉네임과 이메일으로 한정되어 한 사람이 다양한 계정을 사용하는 등의 경우를 추적할 수 없었다. 중고나라나 네이버 측에서는 여러 계정들이 같은 명의의 것인지 확인할 수 있기 때문에 좀 더 정확한 정보 수집이 가능하다. 자세하게는 '같은 명의의 계정 수' 등을 또 하나의 feature로 넣을 수 있을 것이다.
- 사기글의 경우 보통 사기를 친 다음에 성공하면 글을 삭제하기 때문에 글이 올라오는 시점에는 사기글과 정상글의 비율이 비슷해도, 시간이 지나고 크롤링을 할 때는 정상글이 더 많이 남아있을 수밖에 없다. 사기글 데이터 수집에 한계가 있었던 점이 아쉽다. 이에 대한 해결책으로 중고나라와의 협업을 통해 실시간으로 데이터를 받거나, 크롤링을 자주 함으로써 사기글 데이터를 좀 더 많이 모을 수 있을 것이다.

(2) data 분석의 한계

- 에어팟 프로와 같은 특수한 케이스를 제외하면 시세가 시간이 갈수록 확연하게 바뀌기 때문에 다른 품목에 대해서는 "변하는 시세"를 고려하기 위해 별도의 작업이 필요하다. 시계열 데이터를 다루는 방법을 적용하거나 글이 작성된 지 오래될수록 가격에서 일정 비율을 더 빼는 작업이 필요할 것이다.
- 사기글 데이터의 경우 결측치가 있는 경우가 많았다. 합리적인 이유로 결측치를 특정 값으로 채우거나, 결측치가 포함된 data를 지웠지만, 사기글 데이터 개수 부족에 대한 아쉬움이 남는다.
- 가격이나 글 작성 수, 카페 방문 수가 중요한 feature로 포함된 것은 받아들이기에 따라서 당연한 결과로 보일 수 있다.

(3) 제안

- 사기글을 훑어보니 여러 정적인 feature들을 뽑아내는 것도 중요하지만, text mining이나 NLP(자연어 처리)를 통하여 사기글 텍스트의 패턴을 알아내는 것이 모델의 정확도를 높일 수 있다고 판단된다. 단순히 “톡”이라는 단어가 들어있다고 해서 톡으로 거래를 하는 것은 아니며(예-“톡으로 거래하자고 하는 사람은 사기꾼이에요”), “직거래”가 포함된다고 해서 실제로 판매자가 직거래를 하는 것이 아니기 때문에 이 안에 숨겨진 패턴을 파악하는 것이 필요하다.

- 위 연구는 “에어팟 프로 미개봉”에 한정에서 시행되었지만, 다른 품목에 대해서는 또다른 모델을 훈련시켜야 한다. 예를 들면 “에어팟 프로+애플펜슬”을 판매하는 글의 경우 물품 소개를 2번 해야하기 때문에 글자 수가 약 2배 증가할 것이며, 중고 물품을 파는 경우 사용 정도에 따라서 판매 가격이 급감할 것이다. 따라서 물품들을 (우선 전자기기에 한해서) 몇몇 기준에 따라 여러 종류로 clustering을 한 다음에, 각각의 cluster마다 다른 model을 훈련시켜서 적용하는 것이 바람직하다.

(4) 기대효과

- 위 프로젝트의 모델과 앞의 ‘제안’을 종합하여 새로운 모델을 만든다면, 보다 정확한 “사기글 판별기” 역할을 수행함으로써 I.에서 언급했던 대로 중고 온라인 플랫폼에서의 안전한 거래 문화를 정립하고 서비스 질을 향상시킬 것으로 기대된다.