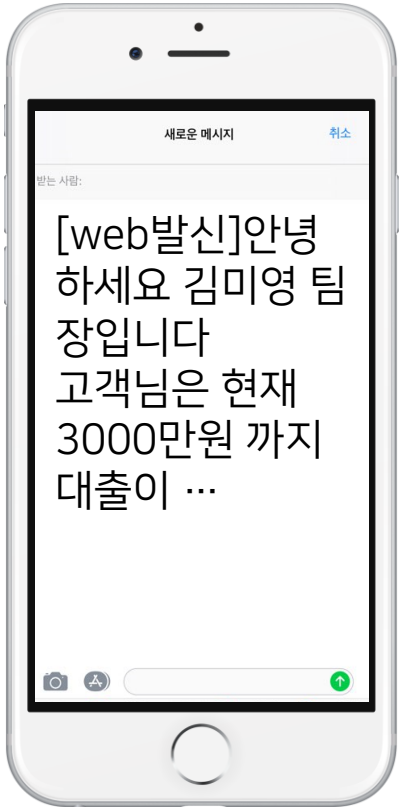


금융권 문자 스미싱 추출 모델

6조
김진만 장시형 김주훈 이은희

주제선정 이유

Smishing 문자메시지 피해현황

2018년 9월 17일 정부부처(방통위, 과기정통부, 금감원) 보도자료

- 명절 안부 인사, 택배 배송 확인 등 스미싱의 기술이 나날이 발전
- '16년 스미싱 피해현황 : 311,911건
- '17년 스미싱 피해현황 : 502,027건 ('16년 대비 61% 증가)
- '18년 8월 스미싱 피해현황 : 161,112건('17.8월 : 274,196 -> '18.8월 : 161,112건 일부 감소)
- BUT, 안부인사, 택배배송, 선물교환권 등을 가장한 지능적인 스미싱 문자로 인하여 피해는 증가할 것으로 예상

2019년 9월 4일 정부부처(과기정통부, 방통위, 금융위, 금감원, 경찰청) 보도자료

- '19.1~7월 피해현황 : 176,220건('18년1~7월 145,093) : 21.5% 증가
- 지인을 사칭한 스미싱이 크게 증가(357.3%, '18.7월 7,470건 -> 19.7월 34,160건)

주제선정 이유

: 금융권 Smishing 문자메시지에 대한 패턴에 대해 호기심이 생겨 Smishing 문자메시지 관련 주제 선정

자연어 전처리의 순서

데이터 전처리(Preprocessing)

토큰화 (Tokenizer)

- 단어를 떼어내는 단계.
- 영문 : NLTK
- 한국어 : KoNLPy

불용어 제거 (Stop word elimination)

- 불필요한 단어들을 제거하는 단계.
- 전체 말뭉치(Corpus)에서 N번 이상 등장하지 않는 단어들을 삭제.
  Why? 별로 중요한 단어라고 생각하지 않기 때문에.(min_df)
- 영문 : Vectorizer 내부에 매개변수로 존재
- 한국어 : 직접 설정

어간 추출 (Stemming)

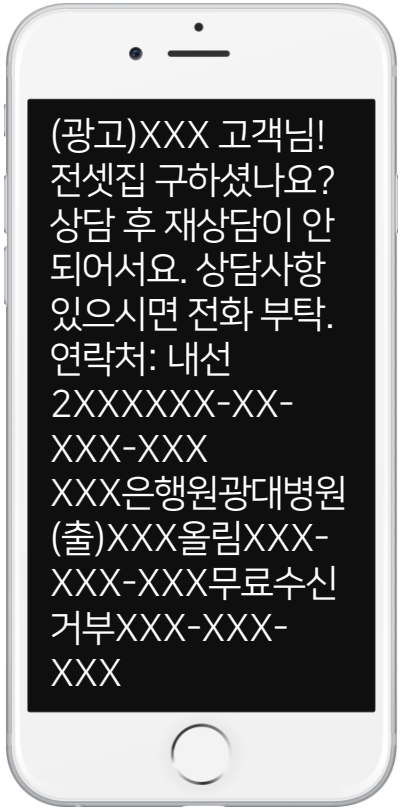
- 단어를 기본형(원형, Root)으로 표현하여 동음이의어로 인덱싱(Indexing)되는 일이 없도록 하는 작업.
- KoNLPy – Okt(stem=True)

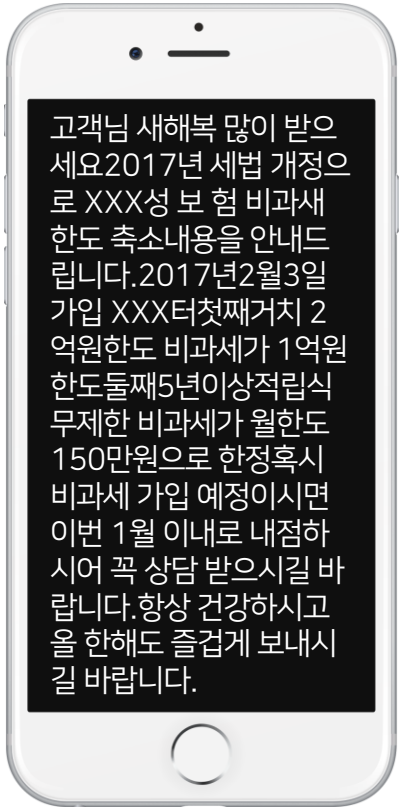
문서 표현 (Representation) = BOW (Bag Of Words)

- Text -> Vector
- 임베딩 (Embedding)
- Multi-hot-encoding
- CountVectorizer
- TfidfVectorizer

Smishing 문자메시지

다음 중 SMISHING 문자메시지를 찾으시오.





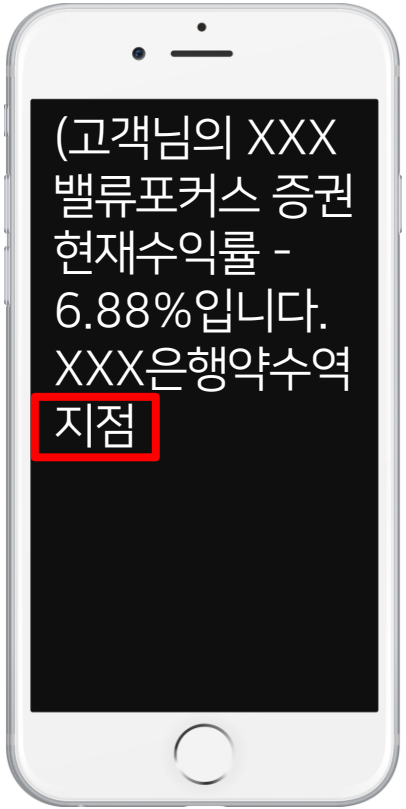
Smishing과 Non-Smishing

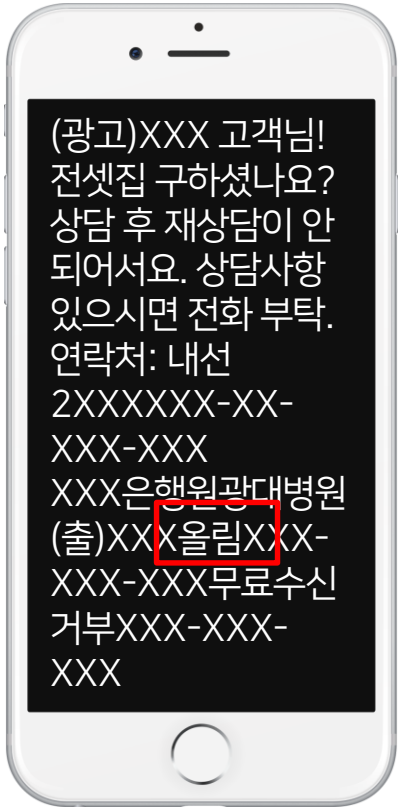
두 문자의 차이점은?

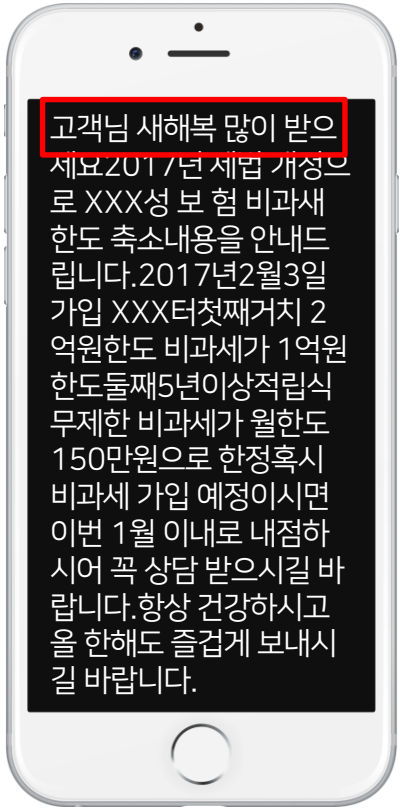
Smishing	Non-Smishing
'레알'과 같은 신조어	지점
사기, 유의하세요	올림
내용 長	인사발령, 지점이전 등 알림문자
'꿀tip' 의도적 한영조합	대리, 팀장, 과장 등 직책표시
'금z ㅣ ' 의도적 오타	새해, 설날 등 인사
카카오톡, 오픈카톡	

Smishing 문자메시지

이 문자들이 Smishing이 아닌 이유

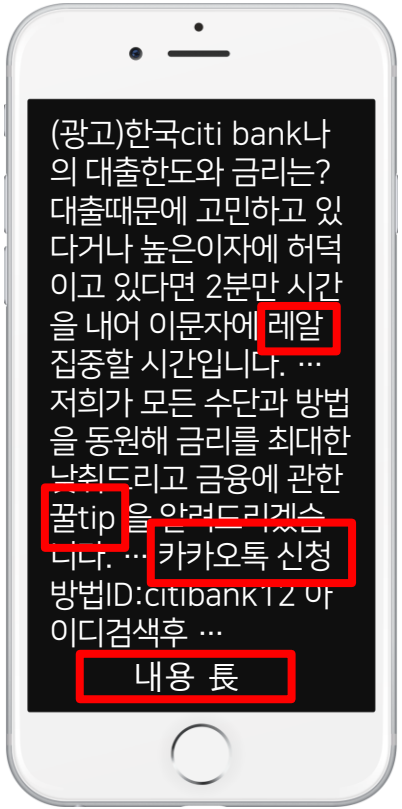


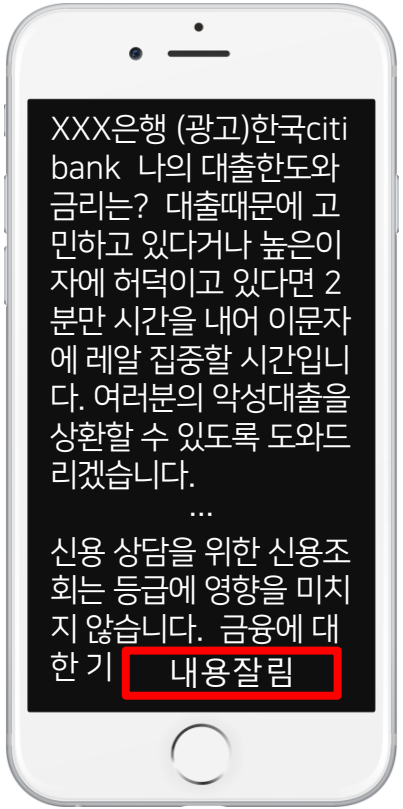




Smishing 문자메시지

Smishing 문자메시지의 예





Smishing과 Non-Smishing

두 문자의 차이점은?

Non-Smishing top30

Smishing top30

Total top30

Tokenize 함수 (2) – Mecab(Linux)

Linux에서 사용할 수 있는 함수 - Mecab

```
졸          VV,*,T,졸,*,*,*,*
면서        EC,*,F,면서,*,*,*,*
봤          VX+EP,*,T,봤,Inflect,VX,EP,보/VX/*+았/EP/*
다          EF,*,F,다,*,*,*,*
왜          MAG,문장부사|양상부사,F,왜,*,*,*,*
이리        MAG,성분부사|공간부사,F,이리,*,*,*,*
평점        NNG,정적사태,T,평점,*,*,*,*
이          JKS,*,F,이,*,*,*,*
높          VA,*,T,높,*,*,*,*
은          ETM,*,T,은,*,*,*,*
겨          NNG,*,F,겨,*,*,*,*
내용        NNG,*,T,내용,*,*,*,*
도          JX,*,F,도,*,*,*,*
별거        NNG,*,F,별거,Compound,*,*,별/NNG/*+거/NNG/*
없          VA,*,T,없,*,*,*,*
[illegible]
아오        NNG,*,F,아오,*,*,*,*
감수성      NNG,*,T,감수성,Compound,*,*,감수/NNG/*+성/NNG/*
영화        NNG,*,F,영화,*,*,*,*
나          VCP+EF,*,F,나,Inflect,VCP,EF,이/VCP/*+나/EF/*
이거        NP,*,F,이거,*,*,*,*
이걸        NP+JKO,*,T,이걸,Inflect,NP,JKO,이것/NP/*+/JKO/*
본          VV+ETM,*,T,본,Inflect,VV,ETM,보/VV/*+/ETM/*
내          MM,~명사,F,내,*,*,*,*
눈          NNG,*,T,눈,*,*,*,*
이          JKS,*,F,이,*,*,*,*
아깝        VA,*,T,아깝,*,*,*,*
[illegible]
——          UNKNOWN,*,*,*,*,*,*,*
진짜        MAG,문장부사|양상부사,F,진짜,*,*,*,*
아          IC,*,F,아,*,*,*,*
놔          VV+EF,*,F,놔,Inflect,VV,EF,놓/VV/*+아/EF/*
진짜        MAG,문장부사|양상부사,F,진짜,*,*,*,*
절대        MAG,성분부사|정도부사,F,절대,*,*,*,*
보          VV,*,F,보,*,*,*,*
지          EC,*,F,지,*,*,*,*
마          VX,*,F,마,Inflect,VX,VX,말/VX/*
라          EC,*,F,라,*,*,*,*
진짜        MAG,문장부사|양상부사,F,진짜,*,*,*,*
제발        NNG,*,T,제발,*,*,*,*
```

졸면서 봤다 왜이리 평점이 높은겨
내용도 별거 없고 아오 감수성 영화냐
이거 이걸 본 내눈이 아깝다——
진짜 아놔 진짜 절대보지마라제발

감탄사 인식 부정확

문자 이모티콘 인식 못함

Tokenize 함수 (2) – Khaiii(Linux)

Linux 내에서 활용 가능한 패키지 - Khaiii

```
졸 면 서    졸 /VV + 면 서 /EC
봤 다       보 /VV + 았 /EP + 다 /EC
왜 이 리    왜 이 리 /NNG
평 정 이    평 정 /NNG + 이 /JKS
높 은 겨    높 /VV + 은 겨 /EC
내 용 도    내 용 /NNG + 도 /JX
별 거       별 거 /NNG
없 고       없 /VA + 고 /EC
아 오       아 오 /IC
감 수 성    감 수 /NNG + 성 /XSN
영 화 냐    영 화 /NNG + 이 /VCP + 냐 /EC
이 거       이 거 /NP
이 걸       이 것 /NP + ㄹ /JKO
본          보 /VV + ㄴ /ETM
내 눈 이    나 /NP + 의 /JKG + 눈 /NNG + 이 /JKS
아 깝 다    아 깝 /VA + 다 /EC
— —         — — /MAG
진 짜       진 짜 /NNG
아 놔       아 늏 /VV + 아 /EC
진 짜       진 짜 /NNG
절 대 보 지 마 라 진 짜 제 발     절 대 /NNG + 보 /VV + 지 /EC + 마 /NNP + 라 진 짜 제 발 /NNG
```

졸면서 봤다 왜이리 평점이 높은겨
내용도 별거 없고 아오 감수성 영화냐
이거 이걸 본 내눈이 아깝다——
진짜 아놔 진짜 절대보지마라제발

댓글 내부의 대부분 표현 인식 가능

KoNLPy 옵션

한국어 정보처리를 위한 Python용 패키지

konlpy.tag : 형태소 분석 및 품사 태깅을 위한 옵션으로 Kkma, komoran, Hannanum, Okt, Mecab이 있음.

** Mecab은 일본어용 형태소 분석기를 한국어에 적용할 수 있도록 수정한 라이브러리

Hannanum	Kkma	Kkma	Mecab	Okt(Twitter)
하늘 / N	하늘/ NNG	하늘 / NNG	하늘 / NNG	하늘 / Noun
을/ J	을 / JKO	을 / JKO	을 / JKO	을 / Josa
나 / N	날 /VV	나 / NP	나 / NP	나 / Noun
는 / J	는 / ETD	는 / JX	는 / JX	는 / Josa
자동차 / N	자동차 / NNG	자동차 / NNG	자동차 / NNG	자동차 / Noun

konlpy.corpus : 말뭉치 옵션으로 kolaw (한국 법률 말뭉치)와 kobill(대한민국 국회 의안 말뭉치)가 있음.

```
>>> from konlpy.corpus import kolaw
>>> fids = kolaw.fileids()
>>> fobj = kolaw.open(fids[0])
>>> print fobj.read(140)
대한민국헌법

유구한 역사와 전통에 빛나는 우리 대한국민은 3·1운동으로 건립된 대한민국임시정부의
```

nouns, morphs, pos()

KoNLPy 매서드

nouns : 명사 추출, morphs : 형태소 추출, pos : 품사 부착

대한민국헌법

유구한 역사와 전통에 빛나는 우리 대한국민은 3·1운동으로

```
mecab.nouns(c[:40])
```

nouns()

```
['대한민국', '헌법', '역사', '전통', '우리', '국민', '운동']
```

```
mecab.morphs(c[:40])
```

morphs()

```
['대한민국',
 '헌법',
 '유구',
 '한',
 '역사',
 '와',
 '전통',
 '에',
 '빛나',
 '는',
 '우리',
 '대한',
 '국민',
 '은',
 '3',
 '·',
 '1',
 '운동',
 '으로']
```

```
mecab.pos(c[:40])
```

pos()

```
[('대한민국', 'NNP'),
 ('헌법', 'NNG'),
 ('유구', 'XR'),
 ('한', 'XSA+ETM'),
 ('역사', 'NNG'),
 ('와', 'JC'),
 ('전통', 'NNG'),
 ('에', 'JKB'),
 ('빛나', 'VV'),
 ('는', 'ETM'),
 ('우리', 'NP'),
 ('대한', 'VV+ETM'),
 ('국민', 'NNG'),
 ('은', 'JX'),
 ('3', 'SN'),
 ('·', 'SC'),
 ('1', 'SN'),
 ('운동', 'NNG'),
 ('으로', 'JKB')]
```

불용어

1) 은, 는, 이, 가, 을, 를, 께, 서, 와, 과, 고, 의, 보다, 한테, 되도록, 처럼, 에, 위해, 들, 에, 게 : 조사
2) 이상, 금리, 대출, 억, 제한, 개인, 상담, 광고, 보다, 상품, 최대, 희망, 거부, 프로, 안내, 수수료, 공동, 년, 서비스, 무료 : 50:50
3) 리브, 스타뱅킹, 카톡 : 고유명사
4) 은행, 사장, 무료, 수신, 한도, 집, 기록, 신용, 올림, 그동안, 기록, 님, 고객, 고객님, 최대, 카톡, 친구, 여신, 다운, 사기 : 출현 빈도수 높거나 낮음
5) TfidfVectorizer(min_df = 6,
max_df=len(X_train)*0.9,sublinear_tf = True)

** 데이터 일부 추출한 Code

```
train_nsm_list=list(train[train['smishing']!=1].index)
train_nsmishing=random.sample(train_nsm_list, 11750 )
```

** 특수문자, 영어, 숫자는 str.replace로 전처리

```
train['text'] = train['text'].str.replace('[0-9]', ' ').str.replace('[A-z]',' ')
    .str.replace('[-=+,./₩?~!@#$^&*|₩₩₩(₩)₩"₩'₩;:ㅡㅇ%]',' ')
```

이상	834	516
금리	1297	802
대출	1208	701
억	914	527
제한	148	85
개인	683	378
상담	1481	791
광고	1482	791
보다	616	323
%	1582	811
상품	1612	826
최대	774	389
희망	722	352
거부	1391	652
프로	60	28
안내	1624	737
수수료	354	159
공동	36	16
년	1233	502
서비스	1229	487
무료	1317	514

Smishing : Non-Smishing = 50 : 50

BOW

Bag Of Words. Vectorizer

BOW(Text -> Vector)

1) CountVectorizer

   문서 집합에서 단어 토큰을 생성하고 각 단어의 수를 세어 BOW 인코딩한 벡터를 만든다.

2) TfidfVectorizer

   - CountVectorizer와 비슷하지만, TF-IDF(Term Frequency – Inverse Document Frequency) 방식으로 단어의 가중치를 조정한 BOW 벡터를 만든다.
   - 단어의 개수를 그대로 카운트하지 않고, 모든 문서에 공통적으로 들어있는 단어는 문서 구별 능력이 떨어진다고 판단하여 가중치를 축소.

Naïve Bayes

Naïve Bayes의 특징과 종류

Naïve Bayes - 각 특성을 개별로 취급해 파라미터를 학습하고, 각 특성에서 클래스별 통계를 단순하게 취합
- 희소한 고차원 데이터에서 잘 작동하며, 비교적 매개변수에 민감하지 않음
- 훈련과 예측 속도가 빠름

1) MultinomialNB
   - 카운트 데이터(특성이 어떤 것을 헤아린 정수 카운트. Ex) 문장에 나타난 단어의 횟수)
   - 클래스별로 평균의 특성을 계산
2) BernoulliNB
   - 이진 데이터(0또는 1)
   - 각 클래스의 특성 중 0이 아닌 것(== 1의 개수)을 카운트

   ** MultinomialNB와 BernoulliNB모델은 복잡도를 조절하는 매개변수 alpha를 갖고 있다.
   alpha값이 크면 클수록 모델의 복잡도는 낮아진다. But, 해당 모델들은 alpha값에 민감하지 않다.
   즉, alpha값이 성능 향상에 크게 기여하지 않는다.
   보통 0이 아닌 특성이 비교적 많은 데이터셋. 즉, 큰문서에서는 MultinomialNB가 BernoulliNB보다 성능이 높다.
3) GaussianNB

   연속적인 값을 처리하는 모델로, 문서분류에 적합한 모델이 아니다.

Naïve Bayes

MultinomialNB와 BernoulliNB의 성능평가

Naïve Bayes 모델 성능평가

```
ensemble1 = Pipeline([('vect', TfidfVectorizer(min_df = 3, max_df=len(X_train)*0.95,sublinear_tf = True)),
                      ('ensemble',BaggingClassifier(base_estimator=MultinomialNB(), n_estimators=1000, max_samples = 0.5)),
                      ])
```

```
[0.9952381  0.99563492 0.99722222 0.99365079 0.9968254 ]
Mean score: 0.996 (+/-0.001)
```

```
ensemble2 = Pipeline([('vect', TfidfVectorizer(min_df = 3, max_df=len(X_train)*0.95,sublinear_tf = True)),
                      ('ensemble',BaggingClassifier(base_estimator=BernoulliNB(), n_estimators=1000, max_samples = 0.5)),
                      ])
```

```
[0.9984127  0.9968254  0.99880952 0.99761905 0.99801587]
Mean score: 0.998 (+/-0.000)
```

** 해당 대회에서는 Training Data Set과 실제 적용하는 Test Data Set이 따로 있다.
위에서 성능평가를 한 Data Set은 Training Data Set을 cv=5로 나누어 평가를 한 것이고,
대회에서 점수 및 순위를 메기는 Data Set은 Test Data Set이다.

MODEL

Pipeline

```
# Pipeline(Vector + Model) – TfidfVectorizer(Multi-hot-encoding) – BaggingClassifier(Ensemble)
ensemble = Pipeline([('vect', TfidfVectorizer(min_df = 3, max_df=len(X_train)*0.95, sublinear_tf = True)),
                     ('ensemble', BaggingClassifier(base_estimator=MultinomialNB(),
                                                    n_estimators=1000, max_samples = 0.5)),
                    ])
ensemble.fit(X_train, Y_train)
```

** MultinomialNB를 사용한 이유 : 보통 0이 아닌 특성이 비교적 많은 데이터셋. 즉, 큰문서에서는 MultinomialNB가 BernoulliNB보다 성능이 높다.
실제 제출 점수도 MultinomialNB가 88점, BernoulliNB는 82점이 나왔다.

BaggingClassifier : 한가지의 모델로 앙상블 모델을 만들어주는 함수. == RandomForest

** base_estimator : 적용할 model(default = DecisionTree)

** n_estimators : 모형의 갯수(defualt = 10)

** max_samples : 하나의 모형에 들어가는 랜덤한 샘플의 갯수(%). 값이 작을 수록 서로 다른 모델 생성 -> 정확도가 높아짐

MODEL

Pipeline

```
# CV
kfold = KFold(n_splits=5, shuffle=True, random_state=0)
results = cross_val_score(ensemble, X_train, Y_train, cv=kfold, scoring = 'accuracy')
results              # [0.99563492, 0.99642857, 0.99722222, 0.99365079, 0.99722222]
results.mean()   # 0.996031746031746
```

실제 대회 점수

** 사이트 개편으로 인하여 처음 제출한 파일 점수 캡쳐 불가.

** 처음 제출한 파일은 해당 대회에서 제공해준 초급형 코드를 기반으로 한 점수이므로 대략적으로 다음과 같음.

** 0.887752 점은 초급형 코드를 기반으로 매개변수 튜닝 및 불용어를 추가 모델로 가장 최근에 제출한 파일의 점수이다.

감 사 합 니 다