**<'AI야, 진짜 뉴스를 찾아줘!'>**

**팀 [으이유] – 제출 코드 설명 (목차 양식을 기준으로 전체 코드의 Flow에 대한 설명)**

## 1. Library 설치 및 데이터, Library 불러오기 / pos_tagger, tokenizer, pretrained_embedding, model 불러오기

- 학습데이터, 테스트 데이터, 정답제출파일을 로드
- 전체 과정에 필요한 **모듈**(Mecab, Fasttext 등) 설치 및 **Library**를 불러오는 단계
- Pos Tagger와 Tokenizer로서 **Mecab**을 사용

## 2. Feature 구성(1) (형태소 분석 + 전처리)

- Mecab을 이용하여 학습 데이터와 테스트 데이터의 'content' 변수에 대한 **토큰화 및 품사 태깅**을 진행, 해당 결과를 이용하여 '각 content의 특성, title과의 관계, 진짜 및 가짜 뉴스의 특성을 반영'하는 **메타 정보 변수**들을 생성

    Ex) 각 content의 평균 토큰 길이(mean_word_len)

    전체 토큰 중 명사가 자치하는 비율(noun)

    각 content 길이 대비 첫 번째 토큰의 길이 비율(first_word_len)

    content가 괄호 및 중괄호로 끝나는지의 여부(end_word_bracket)

    content의 명사 토큰 중 title에 포함된 것의 개수(same_in_title) 등

**(Glove Embedding을 활용하여 content와 title 간의 임베딩 거리를 나타내는 'distance' 변수 생성시, 사용되는 'glove.txt'는 학습 데이터 및 테스트 데이터와 같은 경로(/content)에 있음을 가정)**

### 3. Feature 구성(2) (Simple Model)

- Logistic Regression(LR) / Stochastic Gradient Descent(SGD) / Random Forest(RF) / Decision Tree(DT) / Fasttext. **다섯 개의 모델을 사용**

- 각 모델에서 계산되는 진짜 및 가짜 뉴스일 **확률을 계산하여 변수로서 추가**. 이후 예측 모형의 학습에 사용 (두 확률의 합이 필연적으로 1이므로 진짜 뉴스일 확률만을 사용)

- TFIDF Vectorizer / Count Vectorizer(analyzer = 'word') / Count Vectorizer(analyzer = 'char'). **세 가지 벡터화 기법을 적용**하여 벡터화 된 'count' 변수를 모델 학습에 사용

(Fasttext는 vectorization 진행하지 않음)

- **Stratified 5-Fold Cross Validation**을 적용. 각 모델 학습 시 벡터화 기법마다의 평균 loss 계산

### 4. 예측 (Ensemble)

- 2, 3 번 단계에서 생성된 **메타 정보 변수와 확률 변수를 사용**하여 **XGBoost 예측 모형** 적합 및 예측 진행

- Simple 5-Fold Cross Validation 을 적용 후 평균 예측 확률 계산

- 예측 확률을 통해 진짜/가짜 뉴스 라벨링을 완료한 후, 학습 데이터 내에서 가짜 뉴스로 명시된 content 가 테스트 데이터에 존재할 경우 예측 결과에 관계없이 가짜 뉴스로 라벨링

(광고성 문구를 확실히 가짜 뉴스로 판별하기 위한 과정)

- 제출용 정답지 파일 생성

* 평가 과정에서의 편의성을 위해 .py와 .ipynd 파일을 모두 첨부하였습니다.