사용자가 읽을 네이버 메일에 대한 예측 알림 시스템

1. 연구 배경

오늘날에는 데이터 저장 장치의 발전과 스마트 기기들의 보급, 세계화 등 여러 다양한 이유로 인해서 수많은 데이터가 어디서든지 쌓이고 있다. 이 수많은 쌓인 데이터들을 데이터 홍수라 한다. 옛날에는 데이터의 양이 적어서 정보를 얻기가 어려웠다면, 오늘날에는 데이터의 양이 너무 많아 광고 정보, 사기 등 함정이 존재하며, 내가 필요한 정보를 바로 찾기 어렵다. 이와 같은 사례는 우리의 삶 속에도 많이 존재한다. 그 중, 나는 네이버 이메일에 집중해 사용자의 받은 메일함을 분석하여 사용자에 맞춰 읽을 것으로 예상되는 메일을 추천하는 시스템을 제안하고자 한다. 개인이 받은 메일함 분석을 통해 쓸모없는 메일은 알림이 가지 않고, 읽을 것으로 예상되는 메일만을 알림을 가도록 하는 시스템이다. 이 시스템으로 얻을 수 있는 기대 효과는 메일함에 있는 수많은 메일을 확인할 필요 없이, 사용자가 원하는 정보만을 손쉽게 얻을 수 있게 된다. 그리고 메일이 많이 와서 내가 원하는 메일을 못 보고 지나가는 경우도 이 시스템의 알림을 통해서 놓치는 중요한 메일이 없게끔 할 수 있을 것이다. 메일에 존재하는 가장 큰 정보는 메일 제목과 같은 텍스트 데이터이다. 이 텍스트 데이터를 머신러닝 모델의 피처로 삼아 사용자가 이 메일을 읽을 것인지에 대한 예측방법을 제시하고자 한다.

2. 관련 연구

사용자 기반 네이버 메일 알림 시스템을 구축하기 위해서는 데이터 수집이 선행되어야 한다. 데이터 수집을 위해서는 크롤링을 진행하는데, 크롤링이란 웹상에 있는 데이터를 추출하는 행위라 한다. 크롤링에는 한 페이지 안에서 원하는 정보가 모두 드러날 때 하는 크롤링인 정적 크롤링과 입력, 클릭, 로그인 등과 같이 페이지 이동이 있어야 보이는 데이터를 수집할 때 사용하는 동적 크롤링이 있다. 네이버 메일 홈페이지의 경우는 한 페이지에 메일이 15개씩 존재하고 더 많은 매일 데이터를 얻기 위해서는 다음 페이지로 넘겨야 하기에 동적 크롤링을 사용한다. 그리고 네이버 메일 홈페이지 같은 경우, 자신의 메일함을 볼 수 있는 특징을 지니기 때문에 접근하기 위해서는 로그인이 필요하다. 이는 웹상에서 컴퓨터가 자동으로 조작해주는 selenium패키지의 driver모듈을 이용하여 진행한다.

추출된 메일 정보에는 발신인과 메일 제목이 텍스트 형태로 되어 있고, 이 데이터를 활용하기 위해서 Open Korean Text(Okt)를 활용하였다. Okt는 의미를 가지고 있는 최소단위의 형

태소로 나눠주는 형태소 분석기이며 이 보고서에서는 명사만을 추출한다. 추출한 명사를 머신러닝 모델에 피처로 사용하기 위해 숫자 형태로 값을 표현해야하는데 이를 해결하는 방법으로 TF-idf 벡터화를 진행한다. TF-idf 벡터화는 각 용어가 전체 문서에서 나타내는 출현 빈도의 역 비율을 표현한 방식이다. TF-idf (d,t) 계산법은 다음과 같다. TF-idf (d,t) = TF(d, t) * ln(N / DF(t)) 이다. TF(d, t) 는 문서 d에서 용어 t의 출현 빈도, N은 전체 문서 수, DF(t)는 전체 문서 중 용어 t를 포함하는 문서 수를 뜻한다. 식에서 ln은 밑이 자연로그 e로 사용되었으나, 임의의 수를 밑으로 사용할 수 있다. 그리고 문서의 용어의 TF-idf 벡터값을 전부 제곱하여 더한 후 해당 문서의 분산을 나눈 값이 1이 되도록 L2 정규화를 진행한다. 이러한 과정을 거쳐 메일 데이터에 포함된 메일 발신자와 제목을 벡터화 시킨다.

데이터의 정답 레이블 (읽음 여부) 값을 확인해보니 0인 값이 14823개, 1인 값이 342개이다. 이는 각 데이터 값이 97.7%, 2.3%로 imbalanced data임을 보여준다. 이 같은 경우 문제가 발생할 수 있는데 예를 들어, 모델이 전부 0으로 예측을 하게 되면 정확도가 97.7%인 모델이 나오게 된다. 또한, 사용자 기반 네이버 메일 알림 시스템 같은 경우는 읽어야 하는 중요한 메일이 왔을 때, 알림을 하지 못한다면 치명적일 것이다. 그렇기에 실제 값이 1일 때 1로 예측하는 경우 / (실제 값이 1일 때 1로 예측하는 경우 + 실제 값이 1일 때 0으로 예측하는 경우)의 값이 커야 한다. 이는 class 1에 대한 재현율(Recall)을 뜻한다. 그리고 시스템의 목적을 위해서 알림이 왔을 경우 실제로 사용자가 읽을 것으로 예측될 메일을 추천해야 하는데 이를 위해 실제 값이 1일 때 모델이 1로 예측한 것 / (실제 값이 1일 때 모델이 1로 예측한 것 + 실제 값이 0일 때 1로 예측한 것)의 값이 커야 한다. 이는 class 1에 대한 정밀도(precision)를 뜻한다. 이를 고려하면 시스템의 성능을 위해 재현율을 중요하게 생각하되 정밀도 또한 무시는 할 수 없다. 이를 위해서는 Under Sampling, Over Sampling 기법 등을 통해 해결할 수 있다. Under Sampling 기법은 다수의 클래스(0)에 속해 있는 데이터 중 무작위로 샘플링을 하는 것이다. Over Sampling 기법은 소수의 클래스(1)에 속해 있는 데이터를 복제하여 증가시키는 방식이다. 이는 과적합의 위험이 존재하긴 한다. 이 보고서에서는 Imbalanced Data 해결을 위해 Over Sampling 기법을 사용한다.

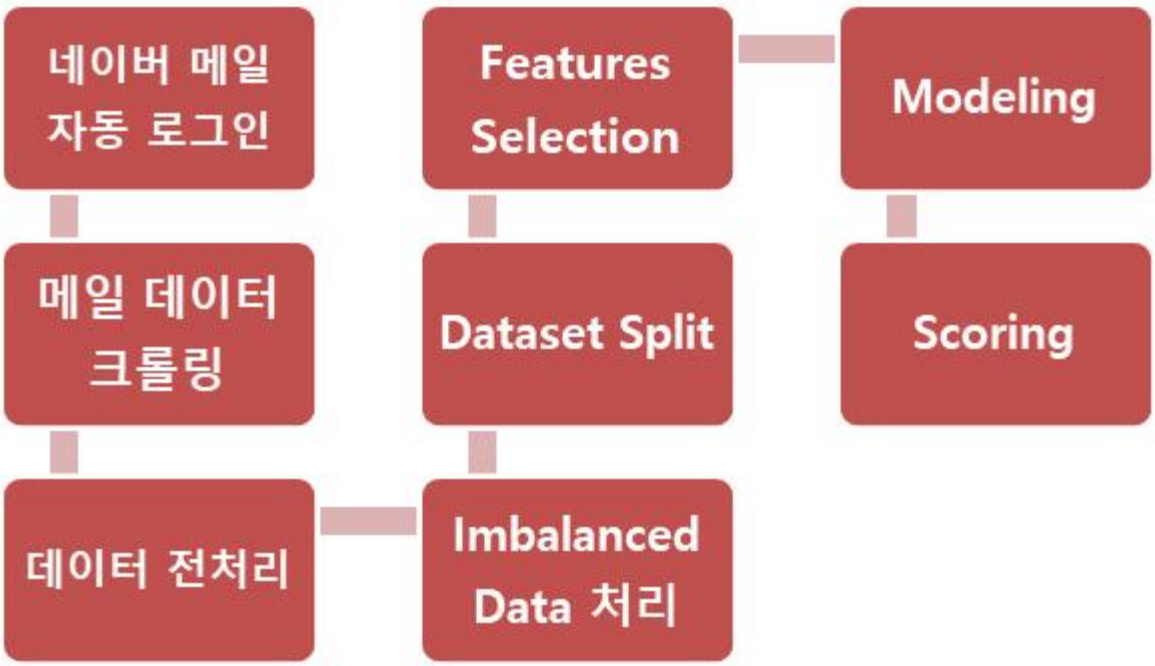
텍스트 데이터를 벡터화 시키다 보니 데이터의 차원 수가 너무 많이 증가하였고, 의미 없는 피처들이 생긴 경우가 있기 때문에 Features Selection을 진행한다. Features Selection은 학습 데이터를 이용하여 정답을 예측하는 데 영향을 끼치지 않는 피처를 제외함으로써 이후 모델링에서 속도 개선 및 메모리 확보의 효과를 얻는다.

모델링은 LGBM (Light Gradient Boosting Machine)과 LR(Logistic Regression)를 사용한다. LGBM 모델은 수직적으로 확장되는 트리 기반의 학습 알고리즘으로 gradient boosting 방식의 모델이다. LR 모델은 회귀를 사용하여 데이터가 어떤 클래스에 속할 확률을 0 ~ 1로 예측하고 그 확률에 따라 클래스를 구분하는 모델이다. LGBM을 사용할 때는 Bayesian

Optimization을 사용해 파라미터값에 따른 검증 데이터 점수를 통해 학습하며 파라미터 최적화 값을 찾고, 찾은 파라미터 값을 이용하여 모델링을 진행한다.

3. 제안방법론

본 장에서는 사용자 기반 네이버 메일 알림 시스템 구현을 위해 다음과 같은 순서를 제시한다.



4. 실험내용

1. 네이버 메일 자동 로그인

이 과정에서는 필자의 네이버 계정을 이용하여 로그인 과정을 실행하였다. 네이버 페이지 같은 경우 자동 로그인을 방지하고 있기 때문에 이를 우회하기 위해서 클립보드 패키지인 pyperclip의 copy모듈을 사용하여 아이디와 비밀번호를 복사한 다음 붙여 넣기 방식을 통해 진행하였다. 그리고 네이버 메일 페이지 로그인 시마다 팝업창이 뜨기 때문에 이를 제거하기 위한 팝업창 닫기에 해당하는 부분을 크롤링해 제거하도록 한다.

2. 메일 데이터 크롤링

메일 데이터 크롤링 항목은 메일마다 있는 메일 읽음 여부, 발신인, 메일 제목, 발신 날짜로 진행한다. 네이버 메일 페이지는 메일함을 넘길 때마다 URL 주소가 바뀌는 형식이 아닌, 메일함을 넘기면 정보가 바뀌는 형식이다. 그렇기 때문에 페이지 내에서 정보를 수집 후, 셀

레니움을 이용하여 페이지를 넘긴다. 이때 한 번에 보여지는 페이지 항목은 10개이므로 마지막 항목에 도달하면 다음 페이지(ex) 10페이지일 경우, 11 ~ 20페이지 칸으로 넘어간다)로 넘길 수 있도록 진행한다. 각각의 크롤링 및 셀레니움 중간에는 time.sleep(1)을 이용하여 오류를 없앤다. 필자의 메일 페이지가 1011페이지까지 존재하여 for 문은 1011번 반복을 진행한다. 시간은 2시간 ~ 2시간 30분이 소요된다. 수집한 데이터들은 데이터 프레임화 시켜준다. 데이터를 수집하는데 오랜시간이 걸리기 때문에 데이터는 저장해두도록 한다.

3. 데이터 전처리

날짜에 대한 정보는 올해에 받은 메일 시각은 2022 년이 제외되어 있기에 이 부분은 따로 "2022-"를 더해준다. 해당 날짜에 메일도 마찬가지로 해당 날짜 부분을 더해준다. 그 다음 pd.to_datetime 을 이용하여 날짜 타입으로 변환 후 연도, 월, 시각 데이터를 추출한다. 읽음 여부에 대한 데이터는 데이터 값이 "읽은 메일", "읽지않은 메일"로 나뉜다. 이는 "읽은 메일"을 1 로 "읽지 않은 메일"을 0 으로 변환시켜 정답 레이블을 만든다. 남은 발신인과 메일 제목 텍스트 데이터는 각각 길이를 이용한 피처를 생성한 후, konlpy 의 Okt 형태소 분석기를 이용하여 명사 부분만을 추출한다. TF-IDF 방식을 이용하여 이를 벡터화 시킨다. 그다음, 벡터화 시킨 두 행렬을 피처명이 중복되지 않도록 조절하여 기존 데이터프레임에 합친다. 기존 원본 데이터 부분은 del()을 통해 없애서 모델링을 할 수 있게끔 데이터프레임을 조작한다.

4. Imbalanced Data 처리

사용자 기반 네이버 메일 알림 시스템이기에 사용자의 메일 데이터 읽음 여부(정답 레이블) 값의 불균형을 확인하고 20 대 80보다 극적인 경우에만 이러한 기법을 사용하도록 한다. 보고서에서 사용된 데이터는 각각 97.7%, 2.3%로 불균형 데이터로 판단하여 데이터 샘플링을 진행하였다. 불균형 데이터에서 리샘플링을 하지 않고 모델링을 진행할 경우, 정확도는 98%가 나왔지만 class 1의 재현율은 27%가 나왔다. 정확도 98%가 나온 이유는 모델이 거의 다 0으로 예측해서 나왔기 때문이다. 보고서의 주된 목적은 메일 알림 시스템이기 때문에 class 1의 재현율이 27%일 경우, 읽어야 하는 중요한 메일을 알리지 않고 넘어갈 수 있기에 문제가 된다. 부가적으로 생각할 것은 정밀도 부분이다. 정밀도가 너무 낮을 경우, 알림이 왔는데 필요 없는 메일이 대다수일 경우가 있다. 그렇기에 너무 낮은 정밀도를 채택한다면 시스템의 성능에 큰 문제가 생길 수 있다. 이에 class 1의 재현율을 높이고자 하는 주된 목적과 class 1의 정밀도가 너무 낮지 않도록 하게끔 하는 부가적인 목적을 가지고 불균형을 해소하기 위해 Under Sampling 기법과 Over Sampling 기법을 진행한다. Under Sampling 기법으로 RandomUnderSampler, Near Miss를 Over Sampling 기법으로 RandomOverSampler, SMOTE, Borderline SMOTE 기법을 진행하였다. 이 중 정밀도와 재현율을 고려하여 정밀도 19%, 재현율 43%의 성능을 보이는 RandomOverSampler 기법을 채택하였다. 여기서 점수를 보기 위해서 사용한 모델은 LGBM 기본 모델이다.

5. Dataset Split

모델링의 점수를 확인하고 학습하기 위해서 Dataset 을 Train set과 Validaion set으로 분할한다. 이때 Train set만을 위해서 채택한 RandomOVerSampler 기법을 통해 리샘플링을 진행한다. 리샘플링 하기 전 Train set의 개수는 10615개이고, 리샘플링을 하고 나서는 20744개가 된다.

6. Features Selection

메일 데이터에서 발신인과 제목의 텍스트 데이터를 벡터화 시켰기 때문에 데이터의 차원이 많이 증가하였고, 그중 의미 없는 피처도 많기 때문에 Features Selection을 진행하여 이후 모델 학습의 속도 향상과 메모리 감소의 효과를 얻을 수 있다. 기존 11083차원에서 Features Selection을 통해 222차원까지 줄일 수 있다. Features Selection은 피처의 중요도가 0 이상인 피처만을 남기는 방식을 사용하였다.

7. Modeling

사용한 Model은 LGBM을 베이지안 옵티마이저로 튜닝한 모델과 LR 모델로 진행하였다. LGBM 튜닝 파라미터는 'n_estimators', 'learning_rate', 'max_depth', 'num_leaves', 'subsample', 'colsample_bytree', 'max_bin', 'reg_lambda', 'reg_alpha' 이다. 기본 파라미터는 조기 종료를 10으로 설정하고, verbose=False를 진행한다. 그리고 점수 계산에는 재현율과 정밀도를 동시에 높일 수 있는 f1_score를 사용하였다. Cross_val_score를 통해 교차 검증을 사용해 더 많은 학습과 검증을 하였고 cross_val_score에서의 파라미터는 cv=2, scoring="roc_auc"를 사용하였다. "roc_auc"를 사용한 이유는 "accuracy", "f1_score", "recall" 등 많은 점수를 사용해보았으나 실험적으로 "roc_auc"가 가장 뛰어난 성능을 보였다. 파라미터 학습은 총 30회 진행하였다. LR는 default 파라미터값을 사용하여 진행하였다.

8. SCORING

점수 계산은 classification_report를 사용하여 각각의 class에 해당하는 점수를 확인하였고, 리샘플링 하기 전 트레인 데이터와 검증 데이터를 이용하여 점수를 계산하였다. 점수 결과는 다음과 같다.

Train score

	precision	recall	f1-score	support
class 0	1,00	0,94	0,97	10372
class 1	0,28	1,00	0,44	243
accuracy			0,94	10615
macro avg	0,64	0,97	0,70	10615
weighted avg	0,98	0,94	0,96	10615

Validation score

	precision	recall	f1-score	support
class 0	0,99	0,93	0,96	4451
class 1	0,14	0,53	0,22	99
accuracy			0,92	4550
macro avg	0,56	0,73	0,59	4550
weighted avg	0,97	0,92	0,94	4550

LGMB classification_report

Train score

	precision	recall	f1-score	support
class 0	0,99	0,69	0,81	10372
class 1	0,06	0,84	0,11	243
accuracy			0,69	10615
macro avg	0,53	0,76	0,46	10615
weighted avg	0,97	0,69	0,80	10615

Validation score

	precision	recall	f1-score	support
class 0	1,00	0,69	0,81	4451
class 1	0,06	0,85	0,11	99
accuracy			0,69	4550
macro avg	0,53	0,77	0,46	4550
weighted avg	0,97	0,69	0,80	4550

LR classification_report

LGBM모델에서는 읽을 것이라고 예측한 것 중 내가 읽을 예정이었던 것은 14%(CLASS 1 정밀도)를 차지한다. 그리고 읽을 예정 중이었던 것 중 읽을 것이라고 예측한 것은 53%(CLASS 1 재현율)를 차지한다. 읽지 않을 것 중 읽지 않을 것이라고 예측한 비율은 93%(CLASS 0 재현율)에 해당한다.

LR 모델에서는 읽을 것이라고 예측한 것 중 내가 읽을 예정이었던 것은 6%(CLASS 1 정밀도)를 차지한다. 그리고 읽을 예정 중이었던 것 중 읽을 것이라고 예측한 것은 85%(CLASS 1 재현율)를 차지한다. 읽지 않을 것 중 읽지 않을 것이라고 예측한 비율은 69%(CLASS 0 재현율)에 해당한다.

5. 결론

➢ 예측 결과

- 기존에 주된 목적으로 CLASS 1의 재현율에 초점을 두었지만 LR모델에서는 CLASS 1 정밀도가 6%로 너무 낮은 점수를 보여 LGBM 모델을 사용하기로 결정하기로 하겠다.
- 피처 셀렉션을 통해 선정된 피처 222개 중 5개를 제외한 217개는 텍스트 데이터를 TF-IDF벡터화 처리한 피처이다. 이를 통해 사용자의 메일에 존재하는 텍스트 분석을 통해서 사용자의 행동을 예측할 수 있음을 알 수 있다.
- LGBM 모델은 53%의 CLASS1 재현율을 보여 내가 읽을 메일을 절반 이상 예측할 수 있다. 그리고 읽지 않을 것을 예측한 93%의 CLASS0 재현율을 보여 어느 정도의 시스템 사용 가능성을 보여준다. 이는 성능을 좀 더 개선시킨다면(한계점을 더 보완한다면) 시스템을 충분히 사용할 수 있을 것이다.

➢ **한계점**

- 보고서에 사용된 필자의 메일 데이터가 불균형 데이터이기에 예측하는 데 어려운 점이 존재한다.
- 어느 정도의 성능을 보이고는 있으나 충분한 성능을 보이지는 못한다. 무엇보다 시스템의 사용자를 위해서 CLASS 1 즉, 읽어야 할 메일을 빠짐없이 알림을 하도록 CLASS 1의 재현율 성능향상이 필요하다.
- 메일 데이터에서 더 많은 정보를 크롤링 해올 필요가 있다.
- 머신러닝 기반의 모델만을 사용하여 딥러닝 기반 모델 사용 시도가 필요하다.