[Kaggle 데이터 분석 포트폴리오]

[1. Kaggle Competition 실전 데이터 분석 – 대표 3건]

1. Random Acts of Pizza

Kaggle Competition 이름:

Random Acts of Pizza
(https://www.kaggle.com/c/random-acts-of-pizza/overview)

Kaggle Competition 설명:

Reddit이라는 해외 커뮤니티의 2010년 12월 8일 ~ 2013년 9월 29일까지 등록된 request 게시물에서 추출된 데이터가 json 파일로 주어진다.

데이터는 다음을 포함한 필드들로 구성되어 있으며, 이들 중 테스트 데이터에 대해 **requester_received_pizza 필드의 값, 즉 request가 성공했는지를 예측해야 한다.**

- ➔ number_of_downvotes_of_request_at_retrieval: 해당 request 게시물의 downvote 수
- ➔ number_of_upvotes_of_request_at_retrieval: 해당 request 게시물의 upvote 수
- ➔ request_number_of_comments_at_retrieval: 해당 게시물에 달린 comment의 개수
- ➔ request_title: 해당 request 게시물의 제목

순위:

129/462 (27.80%)

사용 알고리즘:

1. **학습 데이터에서 사용할 필드를 다음과 같이 지정한다. 대부분 숫자 필드이다.**

- ➔ Request_days_since_first_post_on_raop_at_request
- ➔ Request_number_of_comments_in_raop_at_request
- ➔ Request_number_of_posts_on_raop_at_request
- ➔ Requester_upvotes_minus_downvotes_at_request
- ➔ Requester_upvotes_plus_downvotes_at_request
- ➔ Unix_timestamp_of_request
- ➔ Requester_number_of_posts_at_request
- ➔ Requester_account_age_in_days_at_request
- ➔ Requester_number_of_comments_at_request
- ➔ Requester_number_of_subreddits_at_request

2. **학습 데이터에 대해, 1에서 지정한 필드에 대해서 주성분 분석(PCA, Principal Component Analysis) 방법을 이용하여 차원을 축소한다.**

- ➔ 총 10개의 필드가 있으므로 차원 축소 이전의 차원은 10이며, 이것을 4개의 차원으로 축소한다.

(계속)

사용 알고리즘:

3. **테스트 데이터에 속한 레코드에 대해 k-Nearest Neighbors (kNN) 방법을 이용하여 예측한다. 즉 테스트 데이터의 각 레코드에 대해 Euclidean distance가 가장 가까운 k개의 이웃 레코드를 찾고, 이들의 가중 평균을 이용하여 requester_received_pizza 필드의 값이 True인지 False인지를 예측한다.**

➔ 여기서는 k=130을 적용한다.

➔ 각 이웃 $n_m, m = 1,2,\dots,k$에 대한 가중치 w_m은 다음과 같다.

$$w_m = \frac{1}{d(n_m)} * \frac{N}{N_m}$$

- $d(n_m)$은 n_m과 해당 테스트 데이터 레코드 사이의 Euclidean distance이다.
- N은 학습 데이터의 총 개수, N_m은 이들 중 n_m과 requester_received_pizza 필드의 값이 같은 것의 개수를 의미한다. 이것은 requester_received_pizza 필드의 값에 대한 예측이 편중되는 것을 방지하기 위해 적용한다.

➔ 각 테스트 데이터 레코드에 대한 최종 예측 값은 True=1, False=0으로 하여, 다음과 같이 계산한다.

- $TRUE_m$의 값은 $n_m, m = 1,2,\dots,k$ 중 requester_received_pizza 필드의 값이 True인 레코드에 대한 가중치 w_m의 총합이다.
- $FALSE_m$의 값은 $n_m, m = 1,2,\dots,k$ 중 requester_received_pizza 필드의 값이 False인 레코드에 대한 가중치 w_m의 총합이다.
- 최종 예측 값은 $TRUE_m/(TRUE_m + FALSE_m)$으로 도출한다.

코드가 있는 Github 주소:

https://github.com/WannaBeSuperteur/2020/tree/master/AI/kaggle/2020_08_random_acts_of_pizza (answer.py)

2. Nomad2018 Predicting Transparent Conductors

Kaggle Competition 이름:

Nomad2018 Predicting Transparent Conductors
(https://www.kaggle.com/c/random-acts-of-pizza/overview)

Kaggle Competition 설명:

3,000가지 종류의 화학물질에 대해 다음과 같은 정보가 주어진다.

➔ Spacegroup (해당 물질의 대칭성을 나타내는 label)

➔ 단위 cell에 있는 Al, Ga, O 원자의 총 개수

➔ Al, Ga, In 원소의 상대적 구성비

➔ Lattice vector와 angle

➔ 이때 **테스트 데이터의 레코드로 주어진 600가지의 물질에 대해 Formation energy와 bandgap energy를 예측해야 한다.**

{train|test}/{id}/geometry.xyz 파일에는 해당 레코드가 나타내는 화학물질에 대해, 원자들의 spatial position이 나타나 있다.

순위:

37/878 (4.21%)

사용 알고리즘:

my_convert.py:

테스트 데이터의 각 레코드에 대해,

1. 해당 레코드가 나타내는 물질에 대한 정보가 저장된 {train|test}/{id}/geometry.xyz 파일을 load한다.

➔ 해당 파일에서 추출한 정보에 기반하여, Al, Ga, In, 원자의 전체 원자에 대한 구성비를 전체 원자 개수 기준으로 재계산하여, 각각 percent_atom_al, percent_atom_ga, percent_atom_in라는 필드로 명명한다.

➔ 해당 파일에서 추출한 각 원자의 위치를 기반으로, 각 원자 X에 대해 가장 가까운 원자가 Al, Ga, In, O인 비율을 각각 계산한다. 이 비율을 각각 nearest_X_Al, nearest_X_Ga, nearest_X_In, nearest_X_O라는 필드로 명명한다.

- 이때 X는 Al, Ga, In, O 중 하나이므로, 4x4 = 총 16개의 필드가 생성된다.

➔ 나머지 필드 중 lattice_angle_Y_degree의 경우, 그 필드의 값을 A*15+B로 하여 A, B를 각각 la_Y_degree, la_Y_detailed_degree라는 필드로 명명한다.

- 이때 A는 정수이고 B는 -7.5 이상 +7.5 미만의 값이다. 이런 방식으로 A, B로 분리하는 방법은 항상 유일하다.
- 이때 Y는 alpha, beta, gamma 중 하나이다.

➔ 언급되지 않은 나머지 필드의 경우 원본 데이터를 그대로 이용한다.

2. 위와 같이 변환한 결과를 train_converted.csv (학습 데이터), test_converted.csv (테스트 데이터)로 각각 저장한다.

이때, train_converted.csv, test_converted.csv의 필드 목록은 각각 다음의 train_final, test_final 배열과 같다.

< train_converted.csv의 필드 목록 >

예측 대상, 즉 target variable 필드는 formation_energy_ev_natom, bandgap_energy_ev이다.

```
188     # merge training data
189     train_final = [['id', 'spacegroup', 'number_of_total_atoms',
190                     'percent_atom_al', 'percent_atom_ga', 'percent_atom_in',
191                     'nearest_al_al', 'nearest_al_ga', 'nearest_al_in', 'nearest_al_o',
192                     'nearest_ga_al', 'nearest_ga_ga', 'nearest_ga_in', 'nearest_ga_o',
193                     'nearest_in_al', 'nearest_in_ga', 'nearest_in_in', 'nearest_in_o',
194                     'nearest_o_al', 'nearest_o_ga', 'nearest_o_in', 'nearest_o_o',
195                     'lattice_vector_1_ang', 'lattice_vector_2_ang', 'lattice_vector_3_ang',
196                     'la_alpha_degree', 'la_alpha_detailed_degree',
197                     'la_beta_degree', 'la_beta_detailed_degree',
198                     'la_gamma_degree', 'la_gamma_detailed_degree',
199                     'formation_energy_ev_natom', 'bandgap_energy_ev']]
```

(계속)

< test_converted.csv의 필드 목록 >

```
211       # merge test data
212       test_final = [['id', 'spacegroup', 'number_of_total_atoms',
213                      'percent_atom_al', 'percent_atom_ga', 'percent_atom_in',
214                      'nearest_al_al', 'nearest_al_ga', 'nearest_al_in', 'nearest_al_o',
215                      'nearest_ga_al', 'nearest_ga_ga', 'nearest_ga_in', 'nearest_ga_o',
216                      'nearest_in_al', 'nearest_in_ga', 'nearest_in_in', 'nearest_in_o',
217                      'nearest_o_al', 'nearest_o_ga', 'nearest_o_in', 'nearest_o_o',
218                      'lattice_vector_1_ang', 'lattice_vector_2_ang', 'lattice_vector_3_ang',
219                      'la_alpha_degree', 'la_alpha_detailed_degree',
220                      'la_beta_degree', 'la_beta_detailed_degree',
221                      'la_gamma_degree', 'la_gamma_detailed_degree']]
```

my_extract.py:

1. train_converted.csv와 test_converted.csv 파일을 각각 load한다.
2. 예측 대상(target variable)을 포함한 각 필드의 값들을, 각 필드별로 전체 평균이 0, 표준편차가 1인 표준정규분포로 정규화한다.

➔ 단, spacegroup은 categorical variable이므로 one-hot encoding을 이용한다.

- 이때, 기본적인 방식의 one-hot encoding에서는 그 결과로 0 또는 1을 반환하는데, 여기서는 0에 해당하는 값을 -1로 추가적으로 변환하여 -1 또는 1을 반환한다.

3. 그 결과 중 학습 데이터의 예측 대상 필드를 제외한 나머지 필드의 값, 예측 대상 필드 2개의 값, 테스트 데이터의 예측 대상 필드를 제외한 나머지 필드의 값을 train_input.txt, train_output_0.txt, train_output_1.txt, test_input.txt로 각각 저장한다.

[base]main_mixed.py:

1. train_input.txt, train_output_0.txt, train_output_1.txt, test_input.txt 파일을 각각 로딩한 후 lightGBM, CatBoost 등 최신 딥러닝 모델을 이용하여 학습한다.

➔ 여기서는 CatBoost를 적용하였으며, iteration 횟수는 10이다.

➔ 예측 대상 필드인 formation_energy_ev_natom, bandgap_energy_ev에 대해서 각각 결과를 도출한다.

2. 예측 결과를 final_rmsles.txt 파일로 최종적으로 저장한다.

코드가 있는 Github 주소:

https://github.com/WannaBeSuperteur/2020/tree/master/AI/kaggle/2021_03_nomad2018-predict-transparent-conductors

실행 순서: my_convert.py -> my_extract.py -> [base]main_mixed.py (with CatBoost, 10 iterations)

3. Yelp Recruiting

Kaggle Competition 이름:

Yelp Recruiting

(https://www.kaggle.com/c/yelp-recruiting/overview)

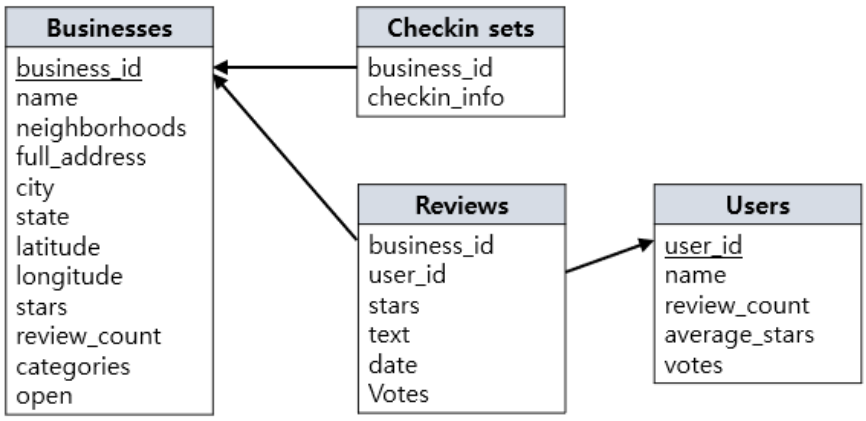
Kaggle Competition 설명:

각각의 Yelp review가 "useful" vote를 얼마나 많이 받았는지를 예측하는 competition이다.

학습 데이터셋의 구성은 다음과 같으며, 각 종류의 학습 데이터는 각각의 JSON 파일에 저장되어 있다.

테이블	# of records	Fields
Businesses	11,537	Business_id, name, neighborhoods, full_address, city, state, latitude, longitude, stars, review_count, categories, open
Checkin sets	8,282	Business_id, checkin_info
Users	43,873	User_id, name, review_count, average_stars, votes (useful, funny, cool)
reviews	229,907	Business_id, user_id, stars, text, date, votes (useful, funny, cool)

이것을 그림으로 표현하면 다음과 같다.



순위:

125/350 (35.71%)

사용 알고리즘:

my_extract.py:

1. business, checkin, review, user 테이블에서 다음의 필드만 이용한다.

Business	Text, stars, date
Checkin	Checkin_info
Review	Text, stars, date
User	Review_count, average_stars

(계속)

2. 각 필드의 값을 다음과 같이 변환시켜서 각각 저장한다.

테이블 이름	필드 이름	변환 방법
Review	text	텍스트의 길이로 변환 후, 표준정규분포로 표준화
	stars	표준정규분포로 표준화
	date	날짜를 숫자로 변환 (연도 * 372 + 월 * 31 + 일) 후 표준정규분포로 표준화
User	review_count	표준정규분포로 표준화
	average_stars	
Business	review_count	표준정규분포로 표준화
	longitude	
	stars	
	latitude	
	open	True이면 1, False이면 -1로 변환
checkin	check_info	텍스트의 길이로 변환 후, 표준정규분포로 표준화

➔ 이때, 테이블 간 상호 참조가 가능하도록 다음과 같이 기본키 또는 외래키의 값을 함께 저장한다.

Business	business_id 필드와 함께 저장
Checkin	business_id 필드와 함께 저장
Review	user_id, business_id 필드와 함께 저장
User	user_id 필드와 함께 저장

➔ 각 테이블에 대해, 저장되는 파일의 이름은 다음과 같다.

	학습 데이터	테스트 데이터
Business	yelp_training_set_business.txt	yelp_test_set_business.txt
Checkin	yelp_training_set_checkin.txt	yelp_test_set_checkin.txt
Review	yelp_training_set_review.txt	yelp_test_set_review.txt
User	yelp_training_set_user.txt	yelp_test_set_user.txt

3. 2에서 저장한 파일을 load한 후, review에 대한 정보가 저장된 파일(~_review.txt)의 각 레코드에 대해 다음을 수행한다.

➔ User_id 필드의 값을 이용하여, user 테이블에서 해당 user_id를 찾아서 대응되는 review_count, average_stars 필드의 값을 학습 데이터의 필드로 한다.

➔ Business_id 필드의 값을 이용하여, business 테이블에서 해당 business_id를 찾아서 대응되는 review_count, longitude, stars, latitude, open 필드의 값을 학습 데이터의 필드로 한다.

➔ 기존에 Review 테이블에 있던 필드인 text, stars, date를 학습 데이터의 필드로 한다.

4. 3의 결과를 학습 데이터는 train_input.txt (입력 데이터), train_output.txt (출력 데이터), 테스트 데이터는 test_input.txt로 저장한다.

(계속)

main.py:

학습 및 테스트 데이터를 load한 후, Deep Neural Network를 이용하여 학습을 수행한다. 총 16번 반복 수행하여 각 회차별로 테스트 입력 데이터에 대한 출력 데이터의 예측 결과(useful vote의 개수)를 저장한다.

- ➔ 여러 번 반복 수행하는 이유는 정확도를 향상시키기 위해서이다.
- ➔ 각 회차의 예측 결과를 저장할 파일명은 test_output_i.txt (i=0,1,...,15) 이다.

write_final_answer.py:

main.py의 예측 결과 파일인 test_output_i.txt (i=0,1,...,15) 를 load한 다음, 각 레코드에 대해 해당 파일에 저장된 대응되는 예측값 (총 16개)의 평균을 구하여 저장한다.

코드가 있는 Github 주소:

https://github.com/WannaBeSuperteur/2020/tree/master/AI/kaggle/2021_04_Yelp_recruiting

실행 순서: my_extract.py -> main.py (with VAL_rate = 0.0) -> write_final_answer.py