|  |
| --- |
| '18년 캡스톤 디자인 프로젝트 |

**프로젝트 결과보고서**

**[Music Recommendation]**

**2017. 12. 18.**

|  |  |
| --- | --- |
| 참여인력(대학) | 백승근(카이스트) |
| 장유진(성균관대) |
| 배지훈(중앙대) |
| 김나원(중앙대) |
| 강준후(경희대) |
| 프로젝트 진행기간 | 2017. 09. ~ 2017. 12. |

목차

**0. 프로젝트 요약 3**

**1. 프로젝트 소개6**

1.1 빅 데이터(Big data)와 프로젝트 동기(Motivation) 6

**2. 프로젝트 목표7**

2.1 활용 가능성 7

**3. 프로젝트 진행과정8**

**4. 프로젝트 세부내용9**

4.1 데이터 분석9

4.2 Neural Network15

4.3 Random Forest21

4.4 Gradient Boosting23

**5. 결론26**

5.1 결론26

5.2 의의26

**6. 참조27**

**APPENDIX28**

Appendix a28

**프로젝트 요약**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **프로젝트 제목** | 프로젝트명 | | | | |
| **구성원** | **이름** | **학교** | **학과** | **전화번호** | **이메일** |
| 백승근 | 카이스트 | 컴퓨터공학 | 010-7550-4329 | baek449@kaist.ac.kr |
| 장유진 | 성균관대학교 | 소프트웨어 | 010-7258-9982 | jangjean96@gmail.com |
| 배지훈 | 중앙대학교 | 컴퓨터공학 | 010-4640-9656 | wnlwnl81@naver.com |
| 김나원 | 중앙대학교 | 컴퓨터공학 | 010-6329-9563 | skdnjskim@gmail.com |
| 강준후 | 경희대학교 | 컴퓨터공학 | 010-9690-7916 | junhukang@khu.ac.kr |
| **지도교수** | Prof. Eric T. Matson | | | | |
| **참여멘토** | Prof. John Springer | | | | |
| **프로젝트**  **필요성 및**  **개요** | 오늘 날 정보통신 분야의 화두 중 하나는 단연 빅 데이터다. 다양한 분야에서 방대한 양의 데이터가 생겨나고, ‘정형화된 데이터’ 부터 ‘비정형 데이터’까지 여러 종류의 데이터들의 처리와 가공 역량의 필요성이 급증하는 시대다. 머신러닝을 통한 여러 응용분야에서 그 효용성은 입증되었으며, 빅 데이터의 활용을 통해 사람의 행동 패턴을 예측하기도 한다. 이는 특히 영리 목적의 기업 입장에서 경쟁력의 필수 요소이다.  이에 본 팀은 소비자의 소비 패턴 분석에 중점을 두고 프로젝트를 시작했으며, Kaggle Competition의 ‘KKBox’s Music Recommendation Challenge’를 주제로 진행했다. 소비자의 개인 정보(나이, 거주지역)와 과거 서비스 기록을 기본 데이터로 특정 노래를 청취한 후 한 달 이내에 재 청취할 가능성을 예측했다.  이 문제에 대해 머신러닝의 ‘Gradient Boosting’, ‘Neural Network’, ‘Random Forest’ 3가지 모델을 사용하였다. 각각의 모델마다 데이터 종류에 따라 가지는 장단점이 있기 때문에 여러 방향에서 문제에 접근하고자 노력했다. | | | | |
| **프로젝트 결과 요약** | Neural Network의 경우 각 모델 요소가 Accuracy에 끼치는 영향을 확인하였다. Neuron과 Layer의 수가 많을수록, 학습이 길어질수록 Accuracy가 상승하며, Dropout과 Logistic Function의 사용은 모델에 부정적 영향을 끼친다. 하지만 Categorical Data의 분석에 있어 Neural Network가 GBM 등 다른 모델에 비해 비교적 효과적이지 못했다.    Random forest 모델 decision tree의 평균 값을 이용해 예측의 안전성을 유지할 수 있었다. 하지만 feature의 수가 많아지면서 classifier 알고리즘의 특징에 의해 다른 두 모델들과 달리 비교적 낮은 정확도를 보였다.    Gradient Boosting 모델에서는 Feature Engineering을 통한 전처리(Preprocessing) 환경에서 다른 두 모델보다 더 나은 효과를 보였다. Categorical 데이터에 대한 복잡한 전처리가 필요 없었고, 은 메모리 사용량과 비교적 짧은 시간으로도 높은 효율성을 보장했다. | | | | |
| **적용 기술** | 개발라이브러리 - TensorFlow, Scikit-Learn  머신러닝 모델 -Light Gradient Boosting, Neural Network, Random Forest  개발 언어- Python, R | | | | |

* **프로젝트 소개 및 목표**

1. **프로젝트 소개**

본 프로젝트는 IITP에서 진행하는 IITP-Purdue 프로그램 내 캡스톤 디자인 프로젝트의 결과이다. 본 팀의 분야는 ‘Big data and Machine learning’ 이었으며 Kaggle사의 Competition 중 ‘KKBox’s Music Recommendation Challenge’를 주제로 프로젝트를 진행하였다. KKBOX는 대만의 음악 스트리밍 서비스를 제공하는 기업으로 이 대회에서 이용자와 노래에 관련된 데이터 셋을 제공한다. 이 데이터 셋을 기반으로 이용자가 어떠한 특정 노래를 청취 했을 때, 그 특정 노래를 한 달 내에 재 청취할 확률을 예측한다. 주어진 여러 종류의 데이터를 가공하고 머신러닝 알고리즘을 통해 예측에 적용한다. 개발 라이브러리로는 ‘TensorFlow’, ‘Scikit-Learn’을 사용하였고 시행 모델로는 ‘Light Gradient Boosting’, ‘Neural Network’, ‘Random Forest’ 가 있으며 개발언어로는 ‘Python’, ‘R’을 사용하였다.

1.1 빅 데이터(Big data)와 프로젝트 동기(Motivation)

현 세대에서 컴퓨터공학 분야의 진보는 빅 데이터(Big data)와 머신러닝(Machine Learning)을 제외하고 논할 수 없다. 그만큼 이미 우리의 현실 생활에서도 깊게 관여하고 있으며 특히 인공지능 분야에서는 그 역량을 여실히 드러내고 있다. 또한 방대한 양의 데이터를 여러가지 머신러닝 알고리즘을 통해 가공, 분석함으로써 기존에는 해결이 어려웠던 문제들에도 접근할 수 있게 되었다.

빅 데이터와 머신러닝의 급속한 발전은 비즈니스 분야에서도 주목받고 있다. 이는 빅 데이터가 세계적으로 통합된 환경에서 상거래에 큰 영향을 끼칠 수 있기 때문이다. 기업들은 빅 데이터 활용을 통해 전반적인 프로세스와 조직 구성도를 개선하며 직접적인 소비자의 행동 패턴을 분석, 예측하기도 한다. 기존에는 무의미했던 데이터를 통하여 새로운 가치를 창출해 내는 것이다.

이에 본 팀은 프로젝트의 주제를 빅 데이터와 머신러닝을 통한 소비자의 소비 패턴 분석에 중점을 두고 시작했다. 기업의 입장에서 소비자의 행태를 예측하고자 했기에 충분한 소비자의 데이터 셋이 필요했고, 기업 측에서 데이터 셋을 제공하는 ‘KKBox’s Music Recommendation Challenge’ 대회에 참가하게 되었다.

1. **프로젝트 목표**

본 프로젝트는 음악 스트리밍 서비스 사의 입장에서 최대 이익을 창출하는 목표의 부분으로서, 이용자(user)와 관련된 방대한 양의 데이터를 통해 소비자의 행동을 예측한다. 데이터 셋을 기반으로 이용자가 어떠한 특정 노래를 청취했을 때, 그 특정 노래를 한 달 내에 재 청취할 확률을 예측한다. 이 예측은 이용자의 새로운 노래와 아티스트에 대한 호감도와 연계될 수 있으며 새로운 이용자에게 효율적인 노래 추천 서비스까지 제공할 수 있게 된다. 이에 각 사용자의 특정 노래 별 재 청취율의 예측 정확도를 최대로 높이는 것을 목표로 한다.

2.1 활용 가능성

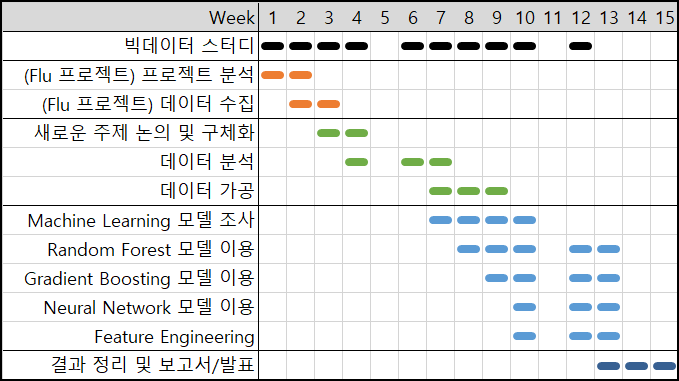
본 프로젝트는 이용자의 개인 정보와 청취 기록을 통해 재 청취율을 예측한다. 소비자 개개인의 경향을 예측한다는 점에서 다수를 위해 일반화된 서비스와 차별성을 둔다. 이러한 개인 맞춤 서비스(Customer Specific Solutions)는 고객과 장기적인 관계를 유지하는 데에 중추적인 역할을 하며 이는 판매 수익률과도 직접적인 연관이 있다.

이용자의 재 청취율을 예측한다는 것은 개개인마다의 특성을 추출할 수 있다는 것을 의미한다. 특정 노래의 재 청취율을 예측함으로써 장르, 작사가, 작곡가 등에 대한 선호도를 알 수 있고 연령별, 지역별 경향 또한 조사할 수 있다. 이는 이용자에게 노래 추천 서비스와 서비스 제공 플랜까지 맞춤으로 제공할 수 있음을 의미한다.

이렇게 고객의 개인 정보와 단기적으로 생성된 서비스 기록으로 소비 패턴을 예측할 수 있다면 이는 고객 관계 관리(Customer relationship management) 시스템에도 영향을 끼친다. 시장과 고객에 대한 이해도가 높아짐으로써 새로운 고객 가치 분석이 가능하게 된다. 또한 잠재 고객, 휴면 고객에 대한 전략적 영업 플랜을 수립할 수 있다.

* **프로젝트 설명**

**3. 프로젝트 진행 과정**

****

(주간 세부 실적은 Appendix A 참고)

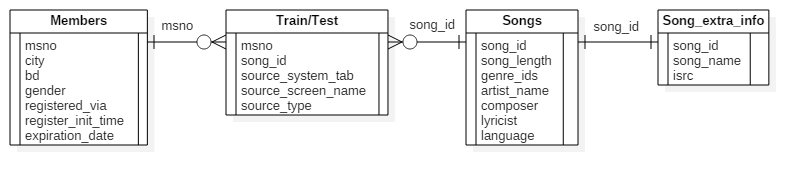
**4. 프로젝트 세부 내용**

4.1 데이터 분석

4.1.1 데이터 구조

본 프로젝트에서 분석할 데이터는 청취 정보를 담고 있는 Train과 Test, 회원 정보(Members), 곡 정보(Songs, Song\_extra\_info)의 파일로 나뉘어서 주어졌으며, 이들 간에는 아래와 같은 Relation이 존재한다.

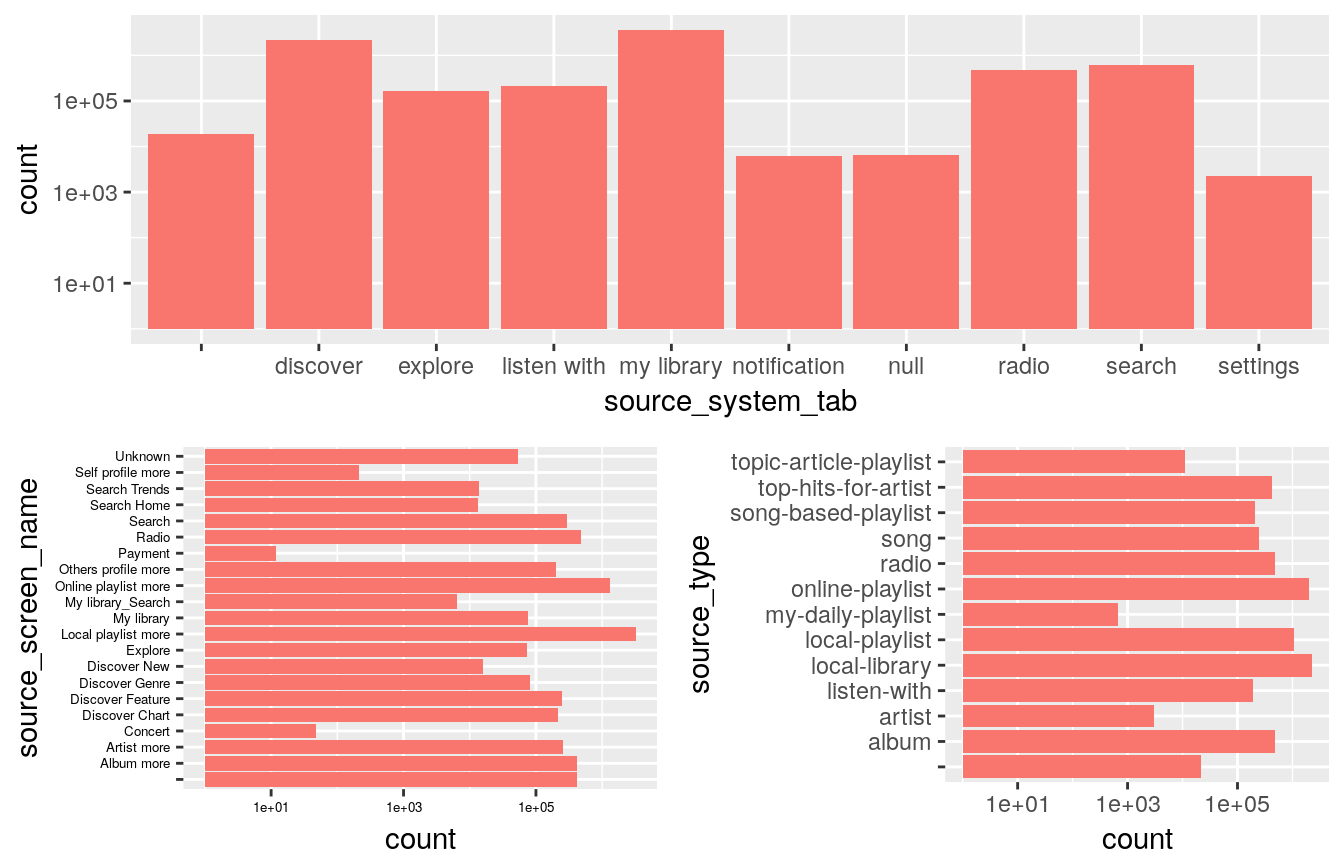
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 데이터 | 의미 | 형식 | 데이터 | 의미 | 형식 |
| msno | 회원 일련번호 | Hashed ID | artist\_name | 가수 | (String)\* |
| city | 도시 | Nominal | composer | 작곡가 | (String)\* |
| bd | 연령(세) | Integer | lyricist | 작사가 | (String)\* |
| gender | 성별 | Binary | language | 언어 | Nominal |
| registered\_via | 가입 경로 | Nominal | song\_name | 곡 제목 | String |
| registered\_init\_time | 회원 가입일 | Date | isrc | ISRC 일련번호 | String |
| expiration\_date | 회원 만료일 | Date | source\_system\_tab | 감상 메뉴 1 | Nominal |
| song\_id | 곡 일련번호 | Hashed ID | source\_screen\_name | 감상 메뉴 2 | Nominal |
| song\_length | 곡 길이(ms) | Integer | source\_type | 감상 메뉴 3 | Nominal |
| genre\_ids | 장르 번호 | (Nominal)\* |  |  |  |



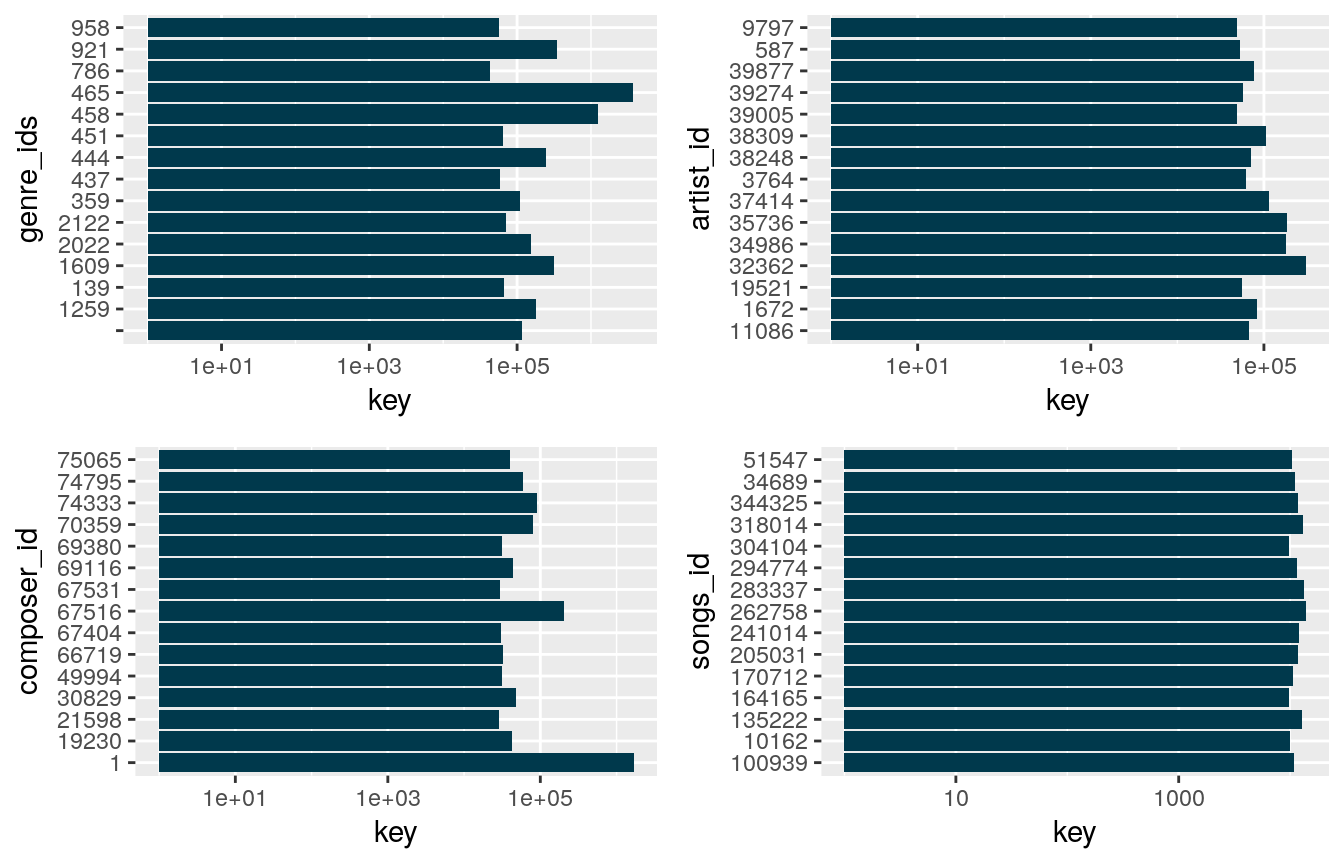
이 데이터는 다음과 같은 몇 가지 중요한 특성을 가지고 있다.

* 데이터의 많은 feature가 분류 데이터(categorical data)이며, 특히 분류 간 순서가 존재하지 않는 nominal data에 해당한다.
* 장르 번호, 가수를 포함한 일부 feature는 바(|)로 구분된 다중 값을 가지고 있다. 이는 주어진 데이터 집합이 제1정규형을 만족하지 않음을 의미한다.
* 회원 정보의 경우 데이터가 완전하지 않은 편이다. 특히 연령의 경우 약 42%의 회원이 정보를 입력하지 않아, 데이터 상에서 연령이 0으로 표시된다.

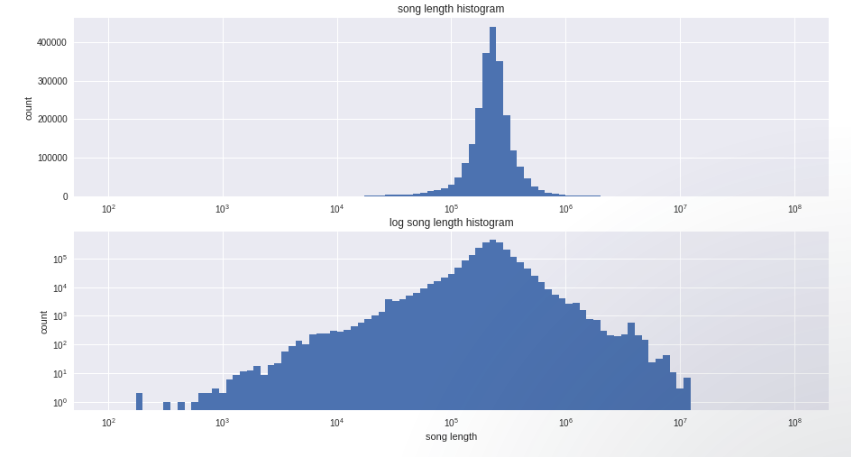
4.1.2. 데이터 개별 분석



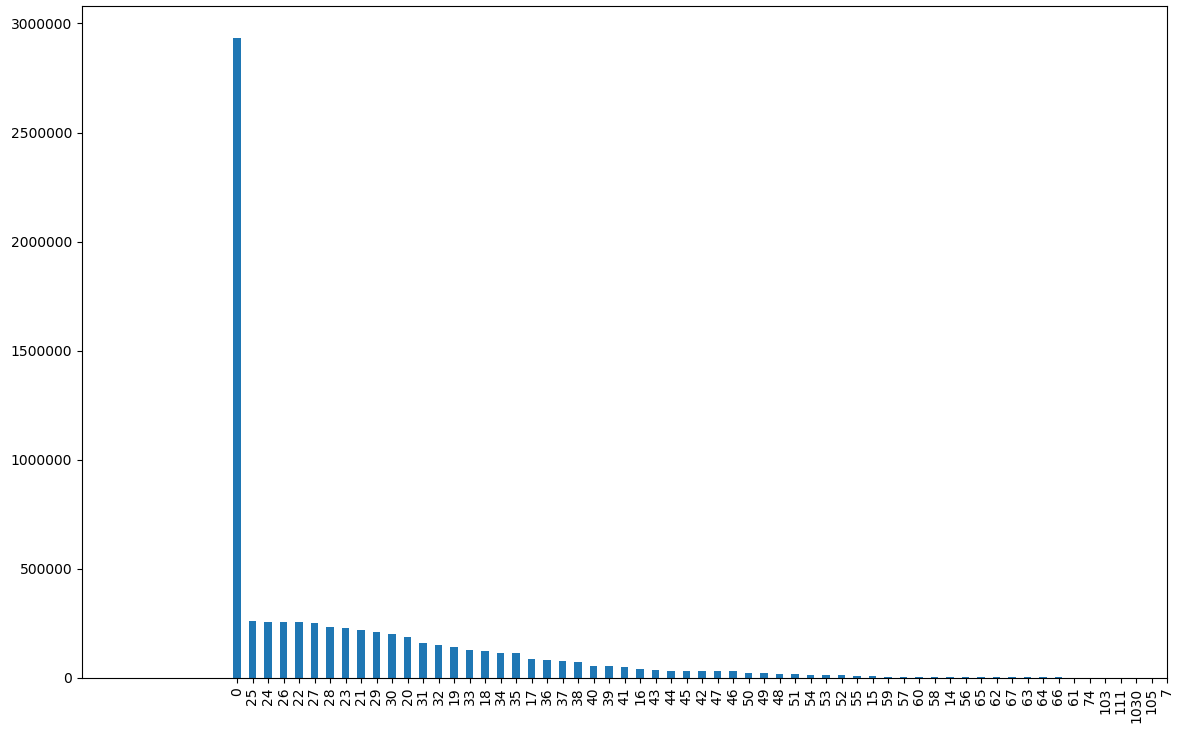
train과 test 데이터에서 볼 수 있는 features에 대한 시각화이다. source\_system\_tab을 통해 시스템에서 선택할 수 있는 menu별 선택 빈도를 알 수 있다. source\_screen\_name을 통해 screen에 나타나는 이름 별, source\_type을 통해 type 별 선택 빈도를 확인할 수 있다.



songs 데이터에서 볼 수 있는 features에 대한 시각화이다. 각 id를 통해 카테고리별 노래분포도를 확인할 수 있다.

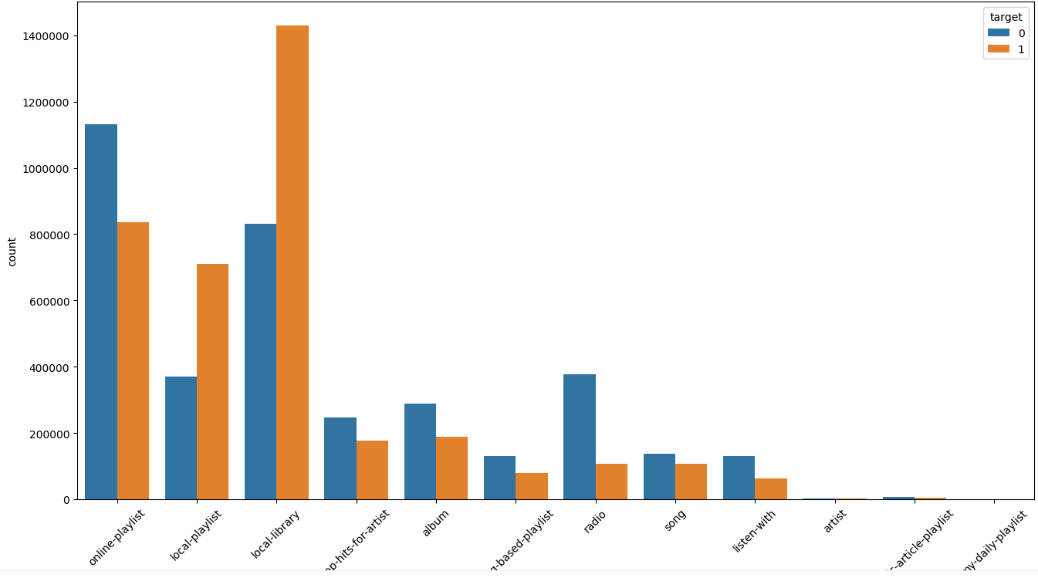


길이에 따른 노래들의 분포를 확인할 수 있는 histogram이다. 노래의 길이라는 특성이 사용자들의 노래 선택에 영향을 미치는지 확인하는데 쓰일 수 있다.

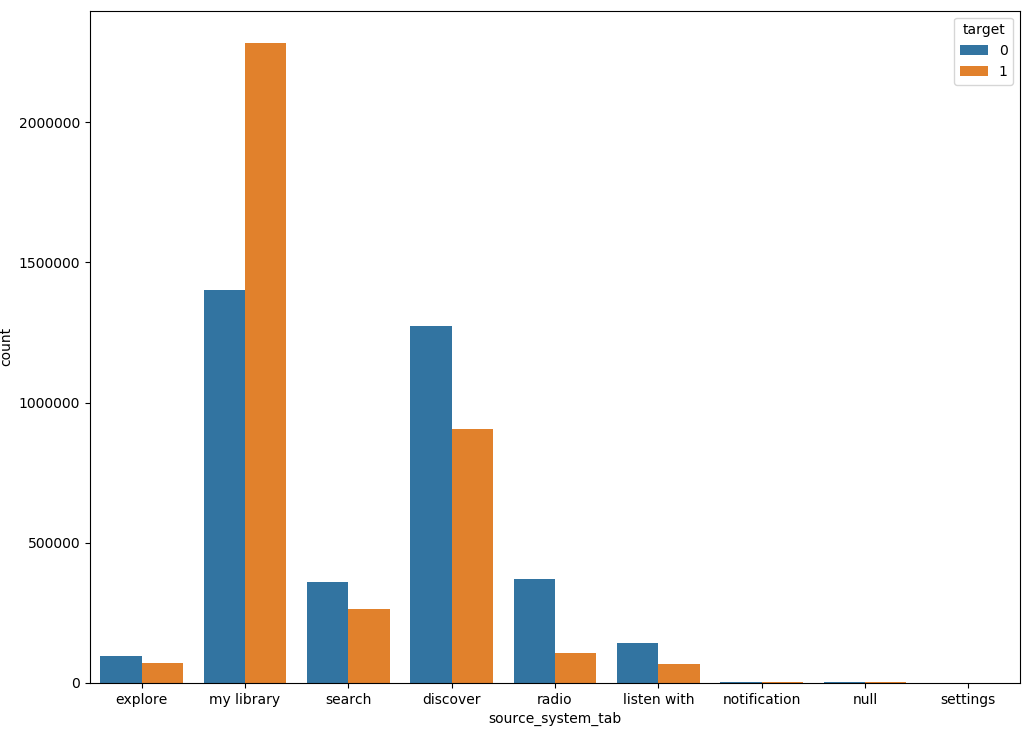


사용자들의 연령대를 확인할 수 있는 그래프이다. 자신의 나이 정보를 입력하지 않은 사용자의 경우 0에 속하므로 0의 값이 가장 크다는 것을 확인할 수 있다. 나이 정보를 입력한 사람들 중에서는 25세 이용자가 가장 많다는 것을 알 수 있다.

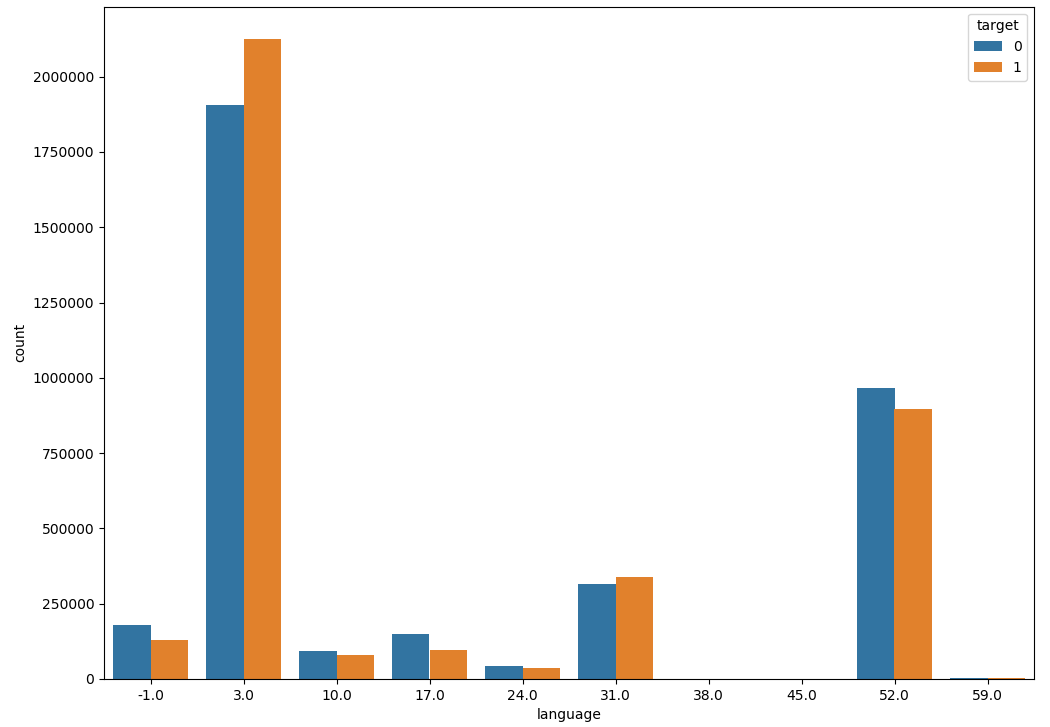
4.1.3 데이터 종합 분석



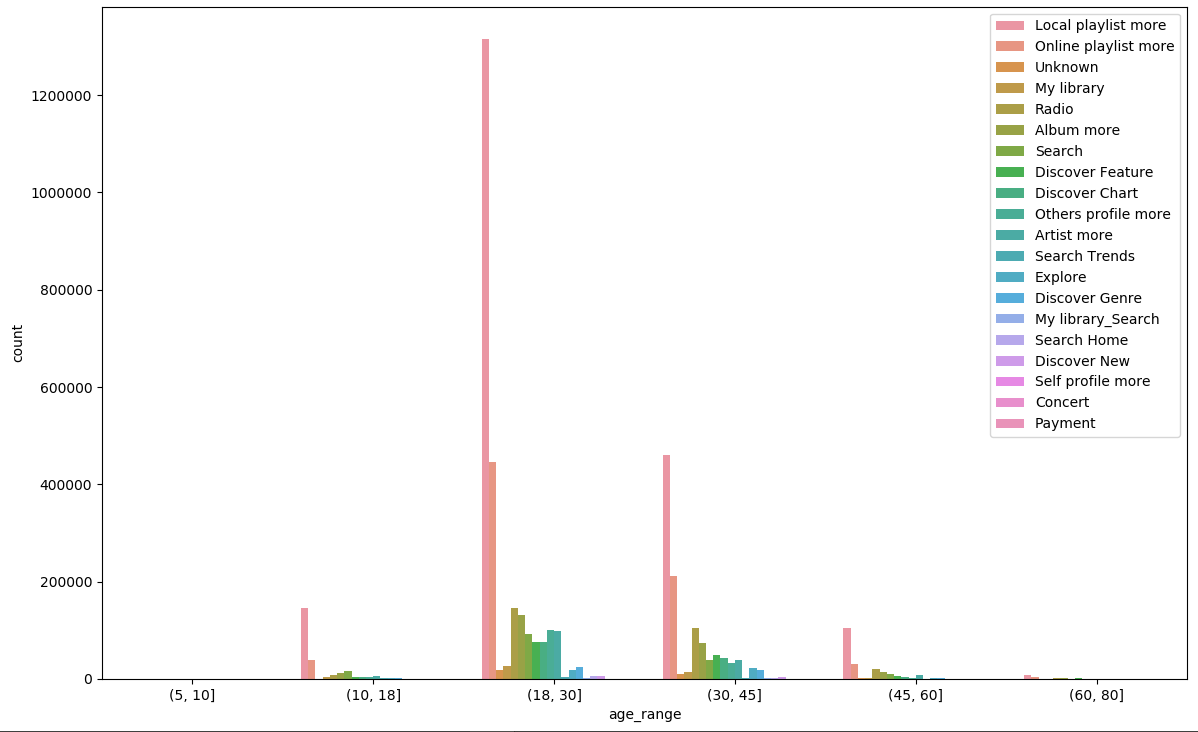
source\_type에 따른 target의 변화를 나타낸 그래프이다. local\_library\_type의 노래들에 대한 반복 재생율이 높은 것을 확인할 수 있다.



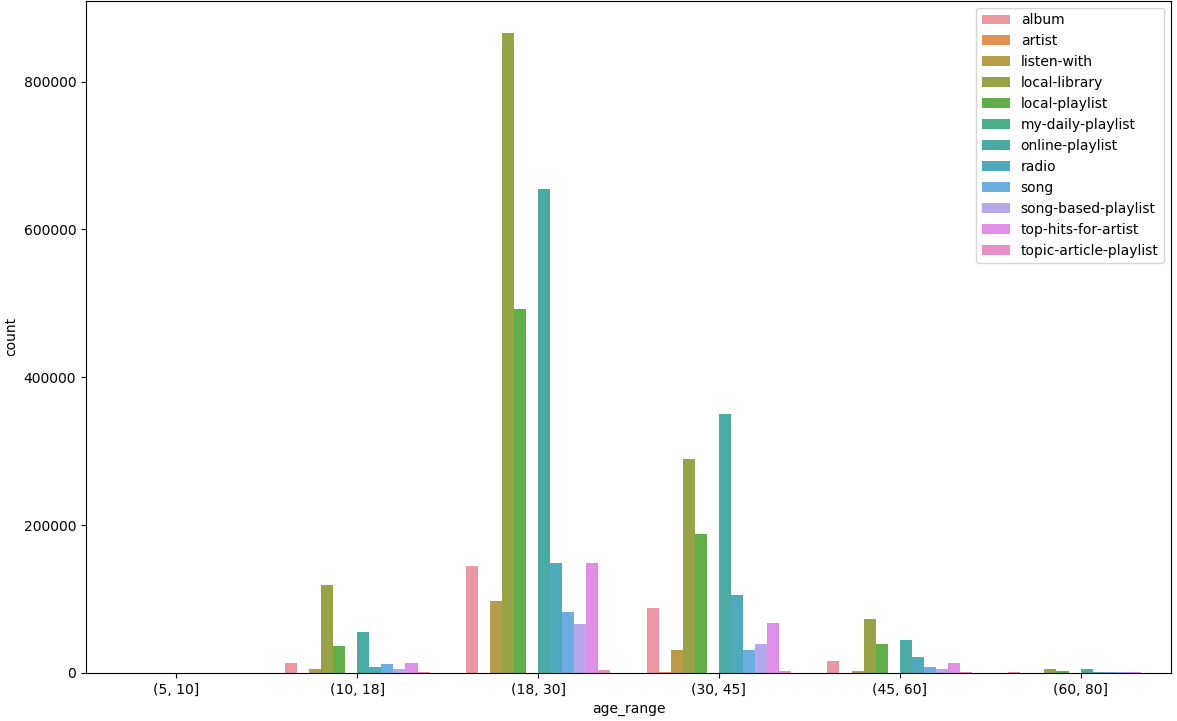
source\_system\_tab에 따른 target의 변화를 나타낸 그래프이다. 이 역시 local library와 같은 맥락으로 my\_library tab에서 높은 반복 재생율을 보임을 확인할 수 있다.



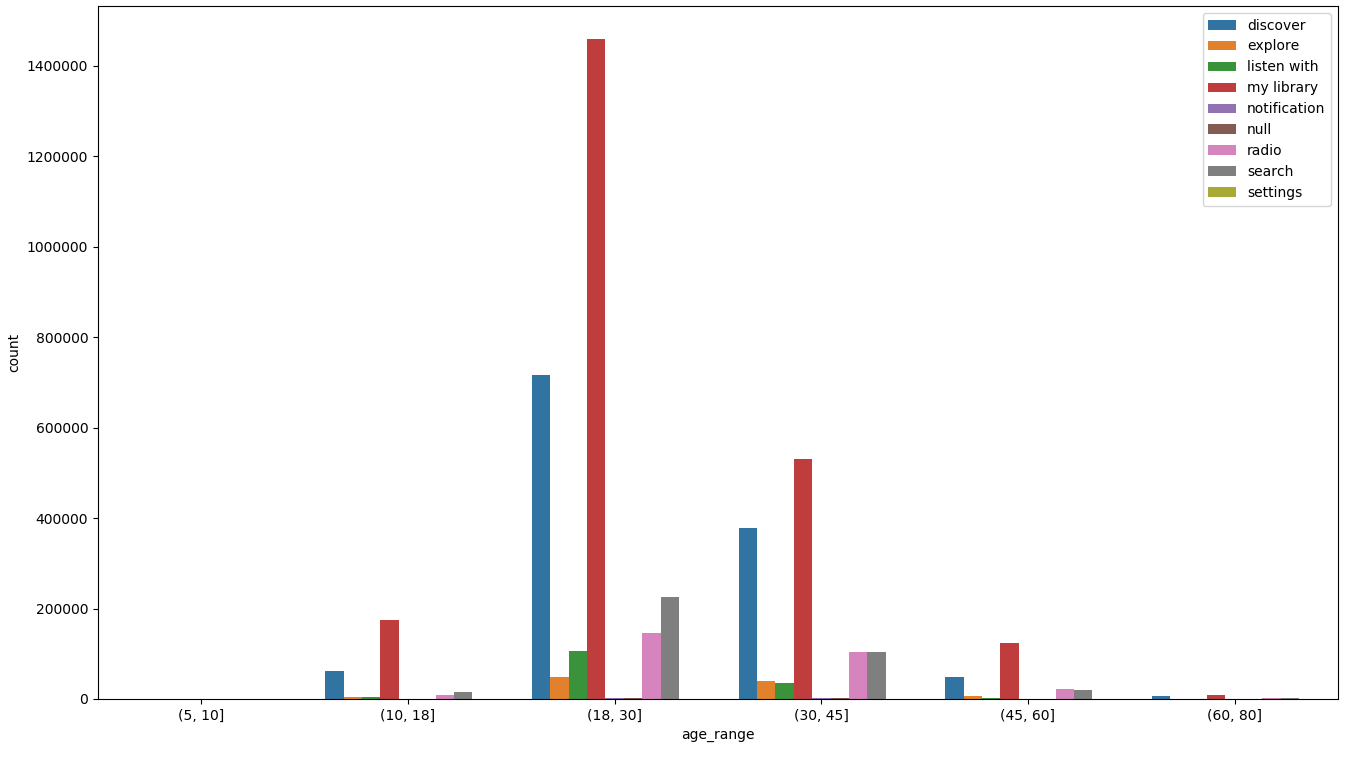
Languages에 따른 target의 변화를 나타낸 그래프이다. 가장 높은 결과를 보이는 언어는 Taiwan, 순서대로 Pop song, K-pop이다.



나이별 선택하는 source\_screen\_name 빈도를 나타낸 그래프이다. 18세에서 30세에 속하는 사람들은 Local playlists more을 가장 많이 선택하는 것을 알 수 있다.



나이별 선택하는 Source\_type 빈도를 나타낸 그래프이다. 18세에서 30세에 속하는 사람들은 local-library를 가장 많이 선택하는 것을 알 수 있다.



나이별 선택하는 Source\_system\_tab 빈도를 나타낸 그래프이다. 18세에서 30세에 속하는 사람들은 my library를 가장 많이 선택하는 것을 알 수 있다.

4.2. Neural Network

인공 신경망(NN)은 최근 재조명받고 있는 머신러닝 기법 중 하나로, 뇌세포의 활동을 모방하여 데이터를 처리한다. 입력 뉴런의 상태 가 주어질 때, 현재 뉴런의 상태는 다음과 같이 계산된다.

여기서 는 각각 weight, bias, 그리고 activation function을 의미한다. Weight와 bias는 처음 무작위로 설정된 후, 학습 과정에서 back-propagation 식을 직접 계산하거나 최적화 알고리즘을 수행함으로써 개선되어 나간다. 이러한 뉴런이 모여 전체 모델을 구성하는데, 편의를 위해 뉴런들의 layer가 주로 사용된다.

최근 연구자들은 이미지 인식 등을 위한 Convolutional NN과 순차 데이터 분석을 위한 Recurrent NN 등 여러 변형된 layer 구성을 활용하고 있다. 그러나 이러한 모델은 각각 입력 layer의 공간적, 시간적 locality를 가정하고 있으므로, 그러한 연관이 없는 여러 feature들을 분석하는 우리 프로젝트에는 적절하지 않다. 이 프로젝트는 layer간의 모든 뉴런이 연결된 dense network 구성을 사용하였다.

Tensorflow와 Theano 등의 몇몇 파이썬 라이브러리가 NN 구현에 초점이 맞추어져 있다. 이 중 GPU를 활용하는 Tensorflow 버전을 이번 프로젝트에 사용하였다.

4.2.1. 데이터 전처리

4.2.1.1. 분류 데이터의 처리

각각의 뉴런은 활성 상태를 숫자로 나타내며 모든 계산이 수치로 이루어지므로, 분류 데이터를 다루기에 NN은 비교적 부적합하다. NN을 nominal data에 사용하기 위해서는, 다음의 과정을 거쳐 데이터를 숫자로 변환해 주어야 한다.

* 성별과 같은 Binary Data는 (male=1, female=0)처럼 바로 변환할 수 있다. 이 경우 Null 값은 단순히 0.5로 매핑하거나, 성별이 알려진 사람 중 남성의 비율을 사용해도 된다.
* Dummy variable

Dummy Variable(Indicator variable)은 특정 조건을 만족하는지의 여부를 0과 1로 나타낸 값이다. 분류 데이터를 Dummy Variable로 나타내기 위해서는 각각의 분류마다 하나씩의 열을 만들어서, 값이 현재 열이 나타내는 분류와 같으면 1, 다르면 0을 부여한다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Genre |  | ID | Genre\_Pop | Genre\_Rock | Genre\_Electronic |
| 1 | Pop |  | 1 | 1 | 0 | 0 |
| 2 | Rock | → | 2 | 0 | 1 | 0 |
| 3 | Pop|Electronic |  | 3 | 1 | 0 | 1 |
| 4 | (null) |  | 4 | 0 | 0 | 0 |

Null(ID 4)과 다중 값 처리(ID 3)의 세부사항은 방법마다 다를 수 있다. Null이나 다중 값이 없어서 각 행에 하나씩의 1이 존재한다면, 이를 Complete Disjunctive Table(CDT)라 한다.

Dummy Variable은 분류 데이터를 인코딩하는 가장 간단한 방법 중 하나이며, 다른 분류 데이터 처리 방법의 기반이 되기도 한다. 서로 다른 분류의 수만큼의 열이 생성되므로 분류의 수가 많은 경우 Feature의 수가 그만큼 많아질 수 있으며, 이를 피하기 위해 상위 n개의 분류에 대해서만 열을 생성하는 등의 방법을 사용할 수 있다.

* Multiple Correspondence Analysis(MCA)

분류 데이터가 여러 개 있는 경우, MCA는 데이터들을 실수로 변환한다. 내부적으로는 분류 데이터에 대한 CDT를 생성한 후 변환을 거쳐 PCA에 넣는 방식으로 계산한다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Source1 | Source2 | Source3 |  | ID | F1 | F2 |
| 1 | My Library | My Profile | My Profile more |  | 1 | 0.88 | 0.79 |
| 2 | Radio | Radio | Listen to | → | 2 | 0.22 | 0.17 |
| 3 | Playlist | Others Profile | Purchase |  | 3 | 0.65 | 0.23 |

MCA의 결과를 좌표로 시각화하면, 데이터의 구조에 대해 추가 정보를 얻을 수도 있다.

청취 경로와 장르를 제외한 다른 분류 데이터(city, registered\_via, language, and artists)는 NN 모델에서 입력으로 사용되지 않았다. 처리 과정에 비해 정확도의 향상이 높지 않은 수준이라 추정했기 때문이다. 상위 32개 장르, 청취 경로를 Dummy Variable로 변환 후 입력으로 사용했다.

4.2.1.2. 데이터 변환 및 정규화

“분류 데이터의 인코딩뿐만 아니라, x값들이 규모가 조정되거나 정규화되어 비슷한 크기를 가질 때 NN은 경험적으로 더욱 효과적이다.”[1] - James McCaffrey, Visual Studio Magazine에서

우리는 입력값을 비슷한 크기(예: 0과 1 사이)로 조정하고 모델의 성능을 향상시키기 위해 데이터의 특성을 반영하고 skewness를 완화시키는 비선형 변환을 사용하였다.

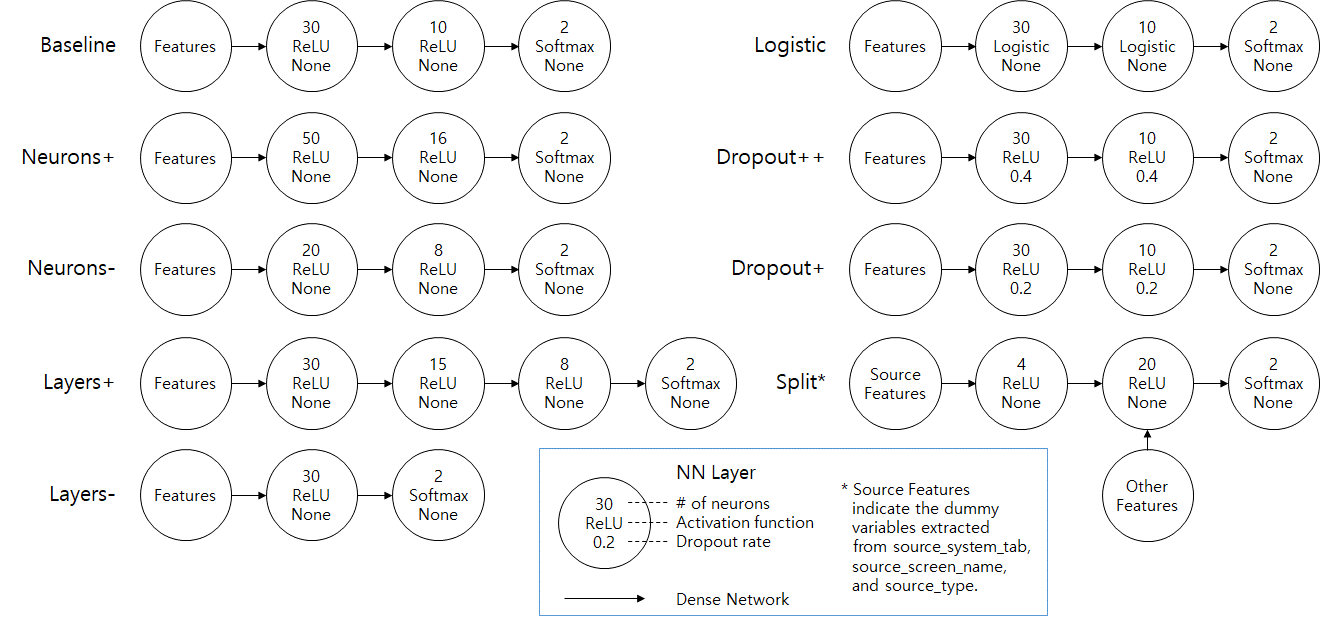
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Normalizer**는 비선형 함수 func를 적용한 뒤, min\_value를 0으로, max\_value를 1로 맞추어 주는 변환 함수를 생성한다. 유효한 변환을 하기 위해서 func는 증가함수여야 한다. | | | | |
| def trimmer(x):  return 0 if x<0 else 1 if x>1 else x  def normalizer(min\_value, max\_value, func, positive=False, unknown=0.5):  return lambda x: unknown if math.isnan(float(x)) or (positive and x<=0) \  else trimmer((func(float(x))-func(max\_value))/(func(min\_value)-func(max\_value))) | | | | |
| Features | func | min\_value | max\_value | Notes |
| bd (age, yr) | Sqrt | 9 | 49 | Adjusting mode(25) into 0.5 |
| song\_length (ms) | Log | 32768 | 1048576 |  |
| member\_duration (s) | Log | 1 | 444873600 | reg\_init\_time – expiration\_date |

|  |  |  |  |
| --- | --- | --- | --- |
| **Decayer**는 현재 시점을 1로 할 때, 지수 함수에 따라 해당 시점의 감소한 값을 계산하는 함수를 생성한다. 이 때 인자로 주어진 반감기가 적용된다. | | | |
| def decayer(current, halflife, unknown=0.5):  return lambda x: unknown if math.isnan(float(x)) else 1 if float(x)>current \  else math.pow(0.5,(current-float(x))/halflife) | | | |
| Features | current | halflife | Notes |
| song\_year (yr) | 2017 | 10 | Data from isrc |
| registration\_init\_time (s) | 1488258000 | 100000000 | current: Feb 28, 2017 |

예를 들어, 최신 곡은 과거의 곡들에 비해 청취 빈도가 더 높을 것이며, 노래의 인기는 보통 지수적 감소 패턴을 보인다. 따라서 곡의 발매 연도를 Decayer(여기에서는 반감기 10년을 사용함)로 전처리할 수 있으며, 이렇게 하면 최신 곡들의 수치가 퍼지는 대신 과거 곡들의 수치가 비교적 비슷한 값으로 바뀌어 최신 곡들에 대해 더 민감한 분석을 할 수 있게 된다.

4.2.2. Network 디자인

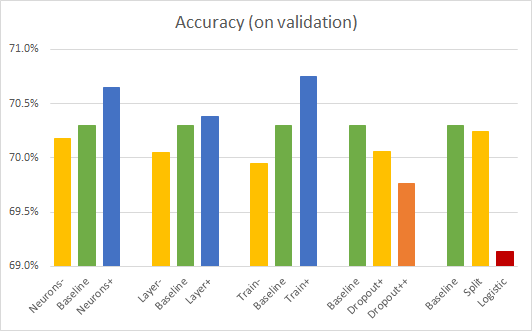
Network 조정에 대한 효과를 알아보기 위해, network를 여러 다른 방법으로 구성하였다. 기준이 되는 network(Baseline)는 dropout이 없는 3-layer network로, ReLU를 activation function으로 사용한다. 뉴런의 수, layer의 수, dropout 비율, activation function 등을 변화시켜 정확도를 확인했으며, input을 분할해 보기도 하였다.



Network의 형태와는 별도로, train 반복 횟수도 조정하였다. Baseline의 경우 50000 step을 사용했으며, 이를 30000(Train-) 및 100000(Train+)으로 조정한 모델도 분석해 보았다.

4.2.3. 결과

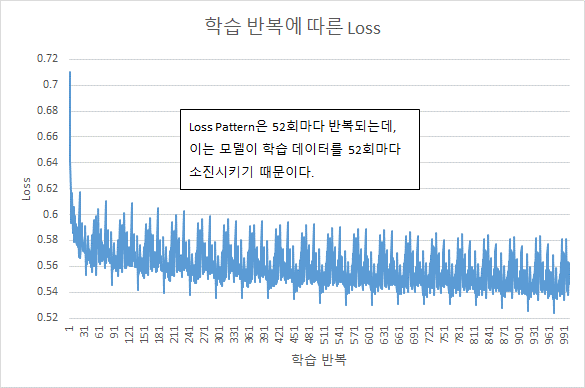
NN으로 분류 데이터를 처리하기 어렵다는 점에서 예상했듯이, 전체적인 validation 결과는 약 70% 정도의 정확도를 보여 LightGBM에 비해 부정확한 것으로 나타났다.



다만 뉴런과 layer의 수, Train 기간 등에 따른 정확도 변화의 경향은 확인할 수 있었다. 뉴런이나 layer의 추가, 긴 학습이 정확도를 향상시킴과 달리, Dropout 기법은 그에 방해가 되는 것으로 나타났다. 데이터를 청취 경로와 그 외의 데이터로 분리해서 처리한 결과(Split), 24개의 뉴런을 사용했음에도 불구하고 Neurons-(28개)와 Layers-(30개)보다 좋은 결과를 얻을 수 있었다.

Kaggle에 업로드한 결과 점수는 validation에서 얻은 정확도보다 낮은 것이 일반적이며, 그 차이는 대략적으로 모델이 데이터를 얼마나 overfitting하고 있나를 나타낸다. Neurons+ 모델로부터 얻은 결과를 업로드하여 62.172%의 점수를 얻었으며, validation 정확도 70.652%와 비교했을 때 그 차이는 약 8.5%p 정도이다.

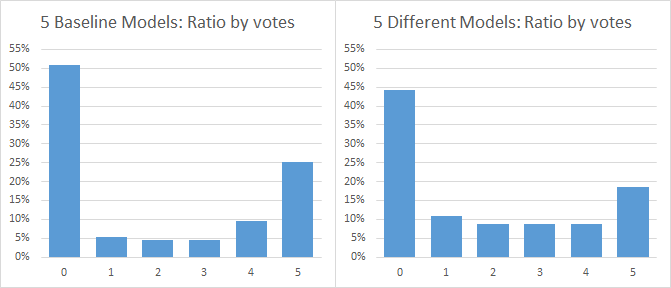
학습을 반복하면서 loss의 변화를 관찰하면 다음 그림(Train+ 모델)과 같은 loss의 감소를 확인할 수 있다.



4.2.4. 투표 방식을 통한 여러 NN 결과의 통합

투표 방식은 가장 간단한 앙상블 모델 중 하나로, 여러 모델에서 생성한 결과를 모아 그 중 가장 많은 모델이 나타내는 결과를 선택하는 방법이다. 투표 방식은 사용된 모델들이 동일한 작업을 하지만 모델의 인자나 Network의 구조가 달라 서로 다른 결과로 수렴하는 상황에서 효과적으로 사용될 수 있다.[2]

동일한 NN 모델(Baseline)을 5회 수행한 결과와 서로 다른 모델(Baseline 2회, Neurons+, Layers+, Split)을 수행한 결과를 이용하여 Voting을 적용하였다.

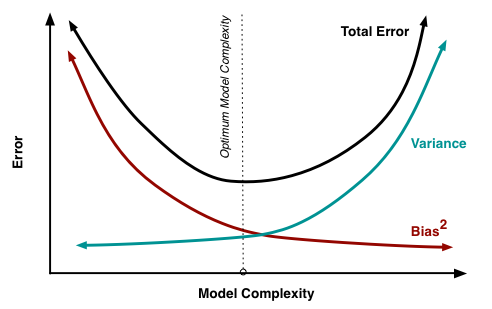
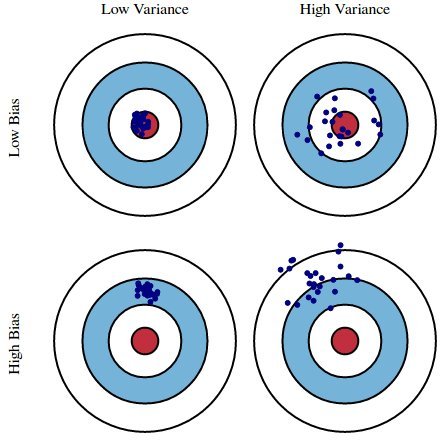


그래프에서 0표와 5표의 비율이 높은 것에서 볼 때, 모델의 수렴 양상이 서로 다른 모델에서조차 비슷하게 나타나 이 프로젝트에서는 투표 방식의 효과가 크지 않은 것으로 확인되었다. 정확도는 유효표의 평균 수치를 기준으로 했을 때, Kaggle 업로드에서 61.262%(Baseline 5회), 61.983%(서로 다른 모델 5회) 정도였다. 이는 8.5%p 정도의 overfitting rate를 고려했을 때 Baseline 단독 수행과 크게 다르지 않은 것으로 보인다.

4.3. Random forest

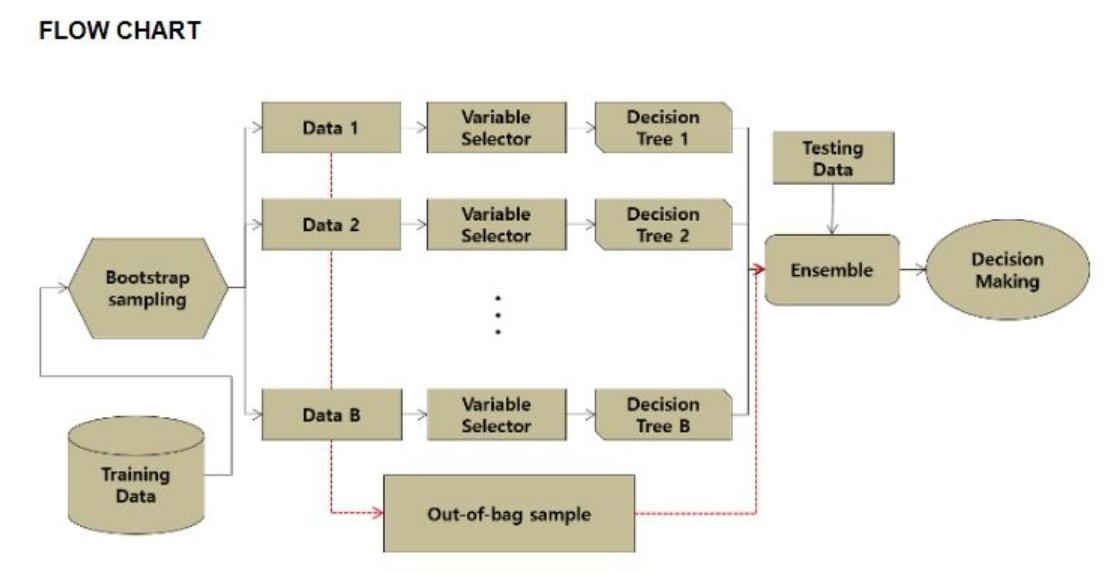
4.3.1 Bagging과 bias-variance trade off

Random forest는 bagging approach 방식을 사용하는 대표적인 머신 러닝 알고리즘이다. Bagging(=Bootstrap Aggregation)이란 주어진 데이터에서 랜덤하게 부분 집합을 샘플링해 N개의 예측모형을 만들어 개별 예측모형에 voting하는 방식이다. Bagging의 장점은 bias-variance trade off를 극복할 수 있다는 점이다. 머신 러닝에서 bias와 variance는 learning error의 주범이지만 서로 zero sum 양상을 띄기 때문에 이 두 값을 낮게 유지해주는 것이 중요하다. (bias가 높으면 예측결과가 실제 결과와 비교해 부정확해지고 variance가 높으면 예측 결과가 어떤 dataset에서는 잘 맞고 어떤 dataset애서는 크게 맞지 안는 등 예측결과의 안전성이 떨어진다.)



4.3.2 Random forest의 알고리즘

Random forest는 불안전성과 같은 decision tree의 단점을 개선하기 위한 알고리즘으로 다수의 decision tree를 결합하여 하나의 모델을 생성하는 방식이다. N개의 bootstrap sample를 생성하고 임의의 bootstrap sample로 N개의 트리를 생성한다. 각각의 앙상블로부터 train classifier를 생성하고 예측 결과를 투표방식으로 선택한다. (Bootstrap은 주어진 train data에서 중복을 허용하여 원해 데이터와 같은 크기의 데이터를 만드는 과정이다. / data를 분류하는 알고리즘 자체를 classifier라고 하며 위의 경우인 decision tree에서는 특정 질문에 대한 응답을 따라가는 방식으로 data를 분류한다.)



4.3.3 bias-variance trade off 극복

Random forest는 bagging approach 방식으로써 bias-variance trade off를 극복한다고 했는데 이를 살펴 보자면 개별 decision tree의 평균값을 이용해 낮은 bias를 유지하고 표본의 크기가 충분하면 표본평균의 분포가 정규 분포를 이룬다는 중심 극한 정리를 이용해 개별 decision tree의 평균값을 사용하는 것으로 variance를 낮게 유지한다. 즉 예측력은 높지만 안전성이 떨어지는 decision tree 여러 개를 평균값으로 사용해 예측력과 안전성을 보장한다.

4.3.4 Random forest의 parameter

지금까지 random forest에 대해 설명했지만 사실 random forest는 black box와 같이 결과가 나오는 과정을 사림이 하나하나 이해할 수는 없다. 하지만 프로그램을 하는데 있어서는 3가지의 hyperparameter만 정의해주면 되기 때문에 사용방법 자체는 용이한 편이다. 먼저 사용할 decision tree의 수와 각각의 decision tree에 사용할 feature의 수를 정해 주어야한다. 마지막으로 전체 data set크기의 몇%로 샘플링을 할지도 정해야 하는데 보통 80%를 사용한다고 한다.

4.3.5 결과

지금까지 random forest를 사용할 때 얻을 수 있는 장점인 예측력, 안전성, 용이성 등을 살펴 봤지만 random forest 방식이 black box와 같다고 한 것처럼 기존 decision tree의 장점인 설명력을 잃는 등의 단점 또한 가지고 있다. 특히 random forest는 data set이 큰 경우 변수가 많은 경우에 권장되지 않는 방법인데 우리가 참여한 WSDM에서는 기본적으로 20여개의 feature 가 제공되었으며 앞에서 살펴본 모델들에 있어서 feature의 수가 많을수록 높은 예측성공률을 보여준 것과 달리 비교적 낮은 결과값을 보여줬다.

4.4 GBM(Gradient Boosting Machine)

4.4.1 Gradient boosting : Gradient Descent + Boosting

부스팅(boosting)이란 단순한 학습기를 결합해서 보다 정확하고 강력한 학습기를 만드는 방식을 의미한다. 정확도가 낮더라도 일단 모델을 만들고, 드러난 예측 오류는 두번째 모델이 보완한다. 이 둘을 합치면 처음보다는 정확한 모델이 만들어지고, 그럼에도 여전히 남아 있는 문제는 다음 모델에서 보완하여 계속 더하는 과정을 반복하는 원리이다. 손실함수는 예측 모델의 오류를 정량화해주는 함수이다. 학습이란 손실함수를 최소화하는 파라미터를 찾는 일이라고 할 수 있다.

Gradient Descent는 손실함수를 파라미터로 미분해서 기울기를 구하고, 값이 작아지는 방향으로 파라미터를 움직이다 보면 손실함수가 최소화 되는 지점에 도달한다. 그 후 Gradient Boosting은 손실함수를 파라미터가 아니라 현재까지 학습된 모델 함수로 미분한다. 정리하면 Gradient Boosting에서는 Gradient가 현재까지 학습된 모델의 약점을 드러내는 역할을 하고, 다른 모델이 그걸 중점적으로 보완해서 성능을 Boosting한다.

4.4.2 데이터의 처리과정

처음에는 데이터의 쓰레기 값이 있으므로 편집하였다. song\_length 중 에서 na인 원소를 평균값으로 채워준다. lyricist, composer, artist 중 에서 na인 원소를 각각 no\_lyricist, no\_composer, no\_artist로 바꾸어 주었다. 또 제공받은 데이터에 있는 각각의 Feature의 타입을 명시화 하였다. 예를 들어 lyricist, composer, song\_id같은 경우는 category타입으로 지정하고 나이 같은 경우는 수치 값인 int타입으로 지정하였다.

다음으로 시간이라는 속성이 노래를 또 들을 확률에 많은 영향을 끼칠 것으로 가정하고 시간과 관련된 Feature를 추가하려고 하였다. expiration\_init\_time, registration\_init\_time을 expiration\_year/month/day와 registration\_year/month/day로 Feature를 세분화하였다. 또 노래의 id에 해당하는 isrc를 이용하여 노래가 발매된 연도를 Feature에 추가하였다. isrc의 5번째부터 7번째의 값이 17보다 크면 2000년도 이전에 발매한 곡이고 작으면 2000년대 이후에 발매한 곡이다.

|  |  |
| --- | --- |
| isrc | song\_year |
| USA01601931 | 2016 |
| GBA7708626 | 1977 |
| GBP53528405 | 1953 |

사용자는 노래가 마음에 들면 해당 곡의 아티스트의 노래를 추가로 들을 확률이 높다고 생각되어 아티스트에 관한 feature를 많이 추가하고 자 하였다. 노래마다 작곡자, 작사가, 아티스트가 몇 명인지 세고 그 각각의 수를 feature에 추가한다. 해당 노래가 피처링 되었는 지 안되었는 지도 새로운 feature로 추가한다. 아티스트와 작곡가가 같은 지 여부, 아티스트, 작사가, 작곡가가 같은 지에 관한 정보도 feature에 추가한다.

마지막으로 노래 길이와 노래 재생횟수를 Feature로 추가하였다. 노래길이가 평균의 절반도 안될 경우 0, 평균 이하일 경우 1, 평균의 1.5배보다 작은 경우 2 나머지는 3으로 값을 같는 feature를 추가한다. 노래가 재생된 횟수를 feature에 추가한다. 해당 노래의 가수의 노래 재생 횟수를 feature에 추가한다.

4.4.3 LGBM의 파라미터

Learning\_rate: 형식은 정수형. 각각의 반복마다 적용되는 축소비율.

Boosting: 형식은 문자열. Boosting 타입.

objective: 형식은 문자열. 학습할 과제를 규정한다. 예를 들면 'regression', 'binary(Classfication)', or 'lambdarank'.

max\_depth: 트리 모델의 최대 깊이.

num\_leaves: 한 트리에 있는 노드 수.

metric: 최적화 할 측정 항목. 'l1'(절대 손실), 'l2'(제곱 손실), 'NDCG', 'AUC'(AUC), 'binary\_logloss'(로그 손실) 및 'binary\_error'(정확도) . 기본값은 'l2'.

verbose : 형식은 bool. True로 두면 평가 셋이 사용되고 평가 진행률을 쓴다.

baggig\_fraction: 형식은 double. 이것은 리샘플링없이 데이터의 일부를 무작위로 선택한다. 훈련 속도를 높이고 과다한 피팅을 처리하는 데 사용할 수 있다.

bagging\_freq: 배깅에 대한 빈도 수. 0은 배깅을 비활성화 함을 의미한다. 예를들어 k라 설정하면 k 반복마다 bagging을 수행하는 것을 의미한다.

bagging\_seed: 행 서브 샘플링의 무작위 시작 시드

feature\_fraction: feature\_fraction이 1.0보다 작으면 각 반복에서 피쳐의 일부를 무작위로 선택한다. 예를 들어 0.8로 설정하면 각 트리를 학습하기 전에 80 % 의 특징을 선택한다.

feature\_fraction\_seed: 열 서브 샘플링을위한 무작위 시작 시드

Max\_bin: 특성 값이 버킷으로 묶일 수 있는 최대 bin수. bin 수가 적 으면 교육 정확도는 떨어지지만 일반 전원은 증가 할 수 있다. Overfitting을 막을 수 있다.

Num\_rounds: 부스팅 반복횟수.

4.4.4 과정

|  |  |  |
| --- | --- | --- |
| 경과 | 추가된 특징 | 정확도 |
| 1 | 제공받은 데이터에 있는 Feature로만 이용. | 62.0% |
| 2 | 제공받은 데이터에 있는 Feature들의 타입을 명시화. | 62.8% |
| 3 | 노래정보에 있는 노래의 연도를 Feature에 추가. | 64.2% |
| 4 | 유저들의 계정 등록일, 만료일을 년/월/일 각각의 Feature로 세분화. | 65.6% |
| 5 | 노래를 만든 아티스트, 작곡가, 작사의 숫자들을 각각의 Feature로 추가. | 66.8% |
| 6 | 노래 길이에 따라 부여한 점수를 Feature에 추가 . | 67.6% |
| 7 | 데이터 상에서 노래가 재생된 횟수를 Feature에 추가.. | 68.3% |
| 8 | 데이터 상에서 해당 가수의 노래가 재생된 횟수를 Feature에 추가. | 68.7% |

4.4.5 결과

Lgbm을 통해서 80.6%의 정확도를 기록하였다. 실제로 제출을 할 시에 68.71%로 정확도가 좀 더 떨어졌지만 전에 시도한 2가지 모델을 포함해서 제일 높은 정확도를 기록하였다. 처음에는 2가지 모델과 비슷한 정확도를 보여줬지만 위에서 언급했듯이 여러 Feature들을 추가하였을 때 정확도가 증가하였다. Learning rate은 0.2, 부스팅 횟수는 100일 때 실제 테스트 데이터에 대한 정확도가 제일 높았다. Learning rate을 낮게 잡았을 때 validation에서 더 높은 정확도를 보였지만 실제 테스트 데이터에서는 정확도가 오히려 낮아지는 overfitting을 보여주었다.

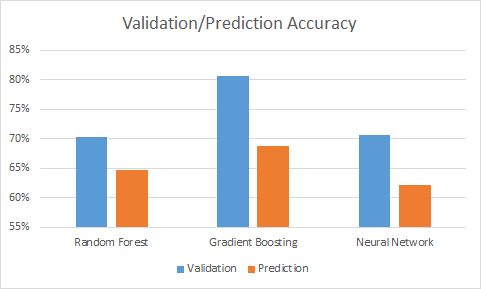
Lgbm은 일반적으로 Feature들을 추가하였을 때 정확도가 증가하는 모습을 보여주었다. 하지만 그 Feature들이 유효한 Feature일 경우에만 정확도가 증가하였다. 위에 언급한 Feature들 이상으로 더 많은 Feature들을 추가할수록 학습시간이 길어지고 메모리가 초과되는 현상이 나타났다. 더 많은 Feature들을 추가하는 데 제한이 있었다

* **프로젝트 결과**

**5. 결론**

5.1. 결론

우리는 다양한 머신러닝 모델 중 3종(Random Forest, Gradient Boostring, Neural Network)을 사용하여 주어진 예측 문제를 해결하고자 하였고, 모델별로 다음의 최적 결과를 낼 수 있었다.



모델 별로 비교하였을 때, GBM이 다른 모델에 비해 예측 성과가 가장 좋았다. Random Forest의 경우 overfitting의 정도(Validation 정확도-Prediction 정확도)가 다른 모델에 비해 낮은 것으로 나타났으며, Neural Network는 분류 데이터의 특성상 예측 정확도가 떨어지는 것으로 확인되었다.

5.2. 의의

팀원 모두가 머신러닝에 대한 배경 지식이 거의 없는 상태에서 프로젝트를 시작하였다. 비록 다른 경쟁자들과 비교해 최상의 결과를 내지는 못했지만 다양한 데이터와 모델을 다루어 볼 수 있는 좋은 기회가 되었다. 특히 모델 아래에 숨겨진 이론과 모델의 세부 구성에 대해 알고, 데이터의 상호 관계나 적합한 분석 방법에 대한 지식을 시행착오를 통해 익힐 수 있었다.

또한 우리가 실제 상거래 환경에서 사용되는 고객 데이터를 분석하고 다른 사람들과 경쟁관계를 유지하며 프로젝트를 진행했다는 점이 매우 흥미로웠다. 이와 같은 경험은 실무 현장에서 실질적인 도움이 될 것이라 생각한다.

**6. 참조**

[1] James McCaffrey. (2014). How to Standardize Data for Neural Networks. Retrieved from <https://visualstudiomagazine.com/articles/2014/01/01/how-to-standardize-data-for-neural-networks.aspx>

[2] E. Alpaydin. (1992). Multiple neural networks and weighted voting. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on Pattern Recognition.

[3] Leo Breiman. (2001). Volume 45, Issue 1, pp 5-32. “Random Forests - Machine Learning.”

[4] Rebecca C. Steorts. (2014). Bagging and Random Forests.

[5] Microsoft. (2017). LightGBM's documentations. https://lightgbm.readthedocs.io/en/latest/

Appendix

Appendix A. 세부 일정표

|  |  |  |
| --- | --- | --- |
| 주차 | 주간 실적 | 비고 |
| 1 | * 담당 교수님과 미팅 * 교수님의 과거 프로젝트에 대한 교육   <Watch Flu Spread: Big Data for Social Good challenge>   * 매 주 미팅 스케쥴 협의 * 빅데이터 스터디 * 빅데이터 정의 * 빅데이터에서의 3Vs * 기업별 동향 * 분석과정 및 방법 * 데이터 얻는 방법 * 응용 사례 |  |
| 2 | * Flu 프로젝트 분석 * 사용 언어: Java * 날씨 이미지 분석을 통한 데이터화: 기온, 풍속, 습도, 강수량 등 * SNS를 통한 데이터 수집: instagram, twitter 등의 게시글 활용 * Google Trend를 통한 데이터 수집: 지역별, 키워드별, 일별 분류 * 데이터 시각화 * D3, Tableau에 대한 공부 * 필요한 새로운 데이터 수집 시도 |  |
| 3 | * Flu 프로젝트 * Flu에 관한 의료데이터, SNS의 위치데이터 등 데이터 수집의 어려움으로 주제 변경 결정 * 새로운 프로젝트 주제 선정: Music recommendation * 데이터 수집이 가능한 주제를 정하기 위해 데이터를 제공해주는 빅데이터 공모전을 검색 후 주제 선정 * 요구사항: 히스토리가 없는 새로운 고객에게 노래 추천하기, 추천했을 때 반복해서 들을 수 있는 노래 추천하기 * ‘노래 선호도는 문화에 영향을 받는다’, ‘노래의 구성요소도 영향력이 있다’, ‘나이, 성별 등의 한 user의 특징도 영향력을 갖는다’ 등의 가정을 세운 후 프로젝트 진행 |  |
| 4 | * Music recommendation 프로젝트 구체화 * 요구사항과 관련되 요소 측정: 다양한 정보를 활용해 한 달 이내에 사용자가 해당 음악을 다시 들을 지 여부 판별 * 사용 언어: Python, R * 데이터 조사 * Kaggle(Big Data 경진대회)을 통해 제공된 KKBox의 자료를 분석 * 음악 정보, 사용자 정보, 음악 접근 정보, 음악 간접 데이터 등을 활용 가능한 데이터로 분류 * 데이터에 적절한 처리를 가하기로 결정 |  |
| 5 | * break | October break of Purdue University |
| 6 | * 데이터 조작 및 분석 * R과 Python을 사용하여 데이터 조작 및 분석: ggplo2 등 여러 라이브러리 사용 * 팀 단위 연구 및 개발 * Data 상호관계에 대한 가설 및 검증: my library - target 관련성 |  |
| 7 | * 다차원 데이터에 대한 Categorical model 탐색 * Categorical PCA * Decision Tree * Random Forest * Ensemble / Boosting * 사용언어 통일 * Python |  |
| 8 | * Data merging * msno등의 공통 feature를 사용하여 제공된 데이터들을 merge * 효과적인 추천을 위해 필요한 새로운 데이터 가공 * Random Forest (in a nutshell) * Tensorflow-GPU 활용 * Feature Range: 3 source columns (3 nominal, 0 numeric) * Parameters: (Records: 7377418, Used for train: 80%, Used for validation: 20%, #Trees: 3, #Ndes: 1000) * Results: (Accuracy: 62.624%, Elapsed Time: 4h 50m) |  |
| 9 | * LightGBM Parameter Tuning * LightGBM model의 Parameter를 조정하여 66.9%의 정확도 기록 * Random Forest * Tensorflow 내의 Random Forest를 사용하는 경우, GPU 사용에도 불구하고 4h 30m 소요 * Scikit-learn 내의 Random Forest를 사용하는 경우, 음악을 감상한 메뉴와 관련된 3가지 feature만 사용하였을 때 57.7%의 정확도를 나타냄 * Super Computer * 슈퍼컴퓨터 시스템이 Python2에 최적화되어 있는 관계로, 호환되지 않는 Python3 라이브러리 및 코드를 직접 수행하기에는 다소 무리가 있음 * Tensorflow-GPU 사용 시, 라이브러리를 개별 컴퓨터에서 build 시 더 좋은 성능을 보임 | [문화체험]  농구 경기 관람 |
| 10 | * LightGBM Parameter Tuning * LightGBM 모델의 Parameter를 추가로 조정하여 67.5%의 정확도를 기록 * Data 전처리 * 음악 장르: Multi-categorical Data인 음악 장르를 분석하여 Train Data 기준 출현 빈도가 높은 32개 장르를 대상으로 Dummy Variable로 변환 * 사용자별 데이터: 재감상 빈도, 많이 재생한 장르, 재감상 빈도가 높은 장르 * Neural Network * TensorFlow가 제공하는 Neural Network 라이브러리를 이해하고, 테스트 모델 구현 * 음악을 감상한 메뉴와 관련된 3가지 feature만 사용한 경우 60%의 정확도로 예측 가능 * 전처리한 장르 정보를 추가하는 경우 2.5%의 정확도 향상 * Node의 개수를 증가시켰을 때 정확도 증가는 미미한 수준이며, Layer를 추가한 경우 역시 동일함 |  |
| 11 | * break | Thanksgiving Break |
| 12 | * 현재까지 성과 및 현황 정리 * LightGBM model, Random Forest model, NN, 데이터 전처리 등 모든 영역에서의 진행상황 파악 및 역할 재 분담 * 단기 목표 수립 * 결과보고서 제출: 12/12 * End of the competition: 12/16 * 결과 발표: 12/19 |  |
| 13 | * Feature 추가 추출 * 개인별 감상한 곡의 평균 요소(발매 연도, 길이 등) * 곡 별 감상한 사람들의 평균 요소(연령 등) * 개인/곡 별 재감상 비율 추출 * LGBM * 새로운 Feature로 인한 모델의 효율 변화는 거의 없었으며, 특히 smooth 처리하지 않은 재감상 비율은 overfitting의 원인이 되는 것으로 추정 * Neural Network * 새로운 Feature 사용 결과 정확도가 0.621까지 향상함 * 다양한 Network에 대해서 시도했으나 큰 효율의 차이는 없었음 * Accuracy 계산 방식에 있어 softmax 처리한 확률을 제공하는 것이 0-1로 분류하는 것보다 좋았음 |  |
| 14 | * 프로젝트 병합 및 정리 * 결과보고서 작성 |  |