웹툰 구독자 수 예측 모델

배건우[0]
제주대학교 전산통계학과
{gunwoof1234@naver.com}

Webtoon subscriber count prediction model

Gunwoo Bae
Dept. of Computer Science and Statistics, Jeju National University

목 차

1. 서론

현재 대한민국에서의 웹툰 시장은 10년 전과 비교할 수 없을 정도로 많이 커졌다. 1996년 한희작의 '무인도'를 시작으로 현재에는 수많은 웹툰들이 한국 시장을 장악하고 있고, 이에 구독자들도 크게 늘어났다.

국내뿐만 아니라 국외에서도 상당한 인기를 끌고 있는데, 특히 만화산업 1등인 일본에서 웹툰 플랫폼 1위, 2위, 4위가 한국 업체일 정도이다. 이로서 웹툰은 k-pop뒤를 있는 하나의 한류를 이끄는 문화라고 말해도 과언이 아니다.[1)]

웹툰 구독자수 예측 모델을 크게 2가지에 활용할 수 있다고 생각한다. 하나는 통해서는 한국 웹툰 시장의 크기를 예측할 수 있다. 다음은 외국웹툰의 구독자 수를 한국웹툰에 구독자 수와 비교하여 외국웹툰의 성장 가능성을 평가할 수 있다. 이를 통해 필자의 모델이 웹툰 시장을 이해하는 하나의 도구가 될 수 있기를 바란다.

2. 본론

2.1 Data collection

캐글에서 'Webtoon Dataset in Korean'를 다운로드 받았다. 데이터의 column은 이것이다.

- id : 열 번호
- title : 웹툰 제목
- author : 웹툰 작가
- genre : 웹툰 장르
- describtion : 웹툰 소개
- rating : 웹툰 평점
- data : 최근 업데이트 날짜
- completed : 웹툰 완결 여부
- age : 구독 제한 나이
- free : 웹툰 무료 여부
- link : 웹툰 사이트 URL

앞의 데이터를 보면 target값인 구독자 수를 나타내는 column이 없을 뿐만 아니라, target값을 예측하기 위한

feature값도 부족해 보인다.

필자는 부족한 feature값을 보완하기 위해 몇 개의 columns를 추가하기로 결정하였다. 추가한 column들이 이것이다.

- previous : 작가의 전 작품의 개수(해당 작품이 나올때를 기준으로 함)
- remake : 원작 유무의 여부
- episode : 웹툰의 에피소드의 개수
- heart : 웹툰이 받은 하트의 개수
- interest_num(target값) : 웹툰의 구독자 수

앞의 새로운 column들을 기존의 데이터에 merge하였다.

previous column은 library를 사용하지 않고 직접 알고리즘을 짜서 구현하였다. 먼저 날자 별로 데이터를 정렬하고, 빈 dictionary를 만든다. dictionary에는 [작가이름(key) : 작품의 개수(value)]를 넣을 것이다. author column의 작가 이름을 dictionary에서 찾는다. dictionary에 없으면 [작가이름 : 0]을 추가할 것이고, dictionary에 이미 있다면 해당하는 작가의 '작품의 개수'에 1을 더할 것이다.

```
# 'previous'열을 만들어서 추가
# 'previous'열 : 작품이 나온 시점에서 전 작품의 수의 개수를 담음
authors_dic={}
previous=[]
for i in data['author']:
  if i not in authors_dic.keys():
    authors_dic[i]=0
  else :
    authors_dic[i] +=1
  previous.append(authors_dic[i])
data['previous']=previous
```

<그림 1> previous column의 핵심 코드

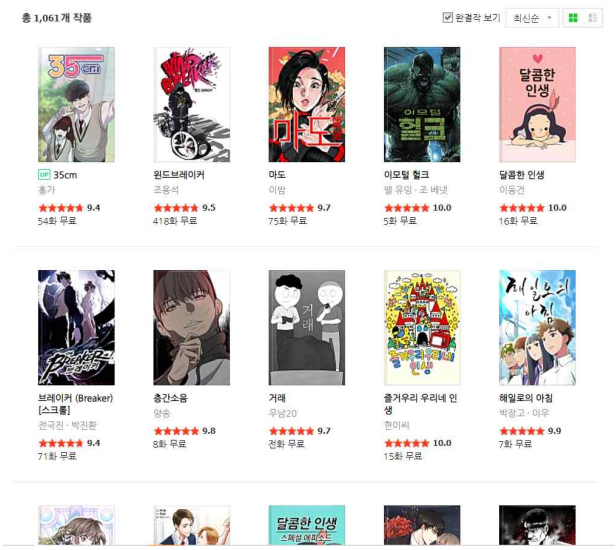
remake column도 library를 사용하지 않고 직접 알고리즘을 짜서 구현하였다. remake된 웹툰을 검색해 보았는데 약 30개 정도였다. 그래서 remake column을 새로 만들고 remake된 30개의 웹툰에만 1을 넣었다. 나머지의 null 값에는 pandas library를 사용하여 null값에 0을 채워 넣었다.

```
# null값에 0넣기
data = data.fillna(0)
```

<그림 2> remake column의 핵심 코드

episode column과 heart column은 Selenium library를 사용하여 네이버 웹툰 페이지를 크롤링을 하여 구현하였다. 처음에는 Beautiful soup library를 사용하려고 했지만, 엉뚱한 내용만 크롤링이 되었다. 원인을 찾아보니 Beautiful soup library는 정적 웹만을 크롤링할 수 있는 도구였다. 그러나 네이버 웹툰 페이지는 동적 웹 페이지이기 때문에 목적에 맞는 데이터를 크롤링 할 수 없었던 것이다. 그래서 동적 웹 페이지를 크롤링 할 수 있는 도구인 Selenium library를 사용하여 크롤링 하였다.

episode 개수를 크롤링 하기 위해 <그림3>의 각 웹툰 사진의 링크로 들어가야 한다. 웹툰의 사진에 hyper link가 걸려 있어서 모든 link를 크롤링하여 list에 저장하였다. 그리고 list의 모든 link를 들어가서 크롤링을 하도록 자동시켰다.



<그림 3> 네이버 시리즈 홈페이지

```
episode={}
for url in urls:
  driver.get(url)
  try:
      wait=WebDriverWait(driver, 5)
      name_path='//*[@id="content"]/div[1]/h2' # 웹툰 이름 x_path
      name_info=wait.until(lambda x: x.find_element(By.XPATH,(name_path))).text
      if '[독점]' in name_info:
        name_info=name_info[:-5]
      episode_path='//*[@id="content"]/h5/strong' # episode의 개수가 적혀있는 x_path
      episode_info=wait.until(lambda x: x.find_element(By.XPATH,(episode_path))).text
      episode[name_info]=episode_info
      print(name_info,episode_info)
  except :
      continue
```

<그림 4> episode column의 핵심 코드

heart column도 앞의 episode와 마찬가지로 웹툰 사진의 링크로 들어 가야한다. 웹툰의 사진에 hyper link가 걸려 있어서 모든 link를 크롤링하여 list에 저장하였다. 그리고 list의 모든 link를 들어가서 크롤링을 하도록 자동시켰다.

```
heart={}
for url in data['link']:
  driver.get(url) # 각 url접속
  try:
    wait=WebDriverWait(driver, 5)
    name_info=wait.until(lambda x: x.find_element(By.XPATH,('//*[@id="content"]/div[1]/div[2]/h2/span[1]')))
    heart_info=wait.until(lambda x: x.find_element(By.XPATH,('//*[@id="content"]/div[1]/div[2]/ul/li[5]/div/a/em')))
    heart[name_info.text]=heart_info.text
    print(name_info.text,heart_info.text)
  except :
      continue
```

<그림 5> heart column의 핵심 코드

interest_num(target값)은 특이하게 웹 페이지에서는 없고, mobile 어플로 들어가야만 있었다. mobile 어플은 pc로 다운받을 수 없어서 어쩔 수 없이 모든 웹툰을 보고 구독자수의 데이터를 기입고 column을 만들었다.

2.2 Data processing

Data processing을 하기에 앞서서 주관적인 판단으로 필요없어 보이는 column들을 지웠다. id, title, author, describtion, date, completed, link는 interest_num(웹툰의 구독자 수)을 예측하는데 도움이 안될 것 같아서 삭제하였다. 삭제하고 남은 column들은 이것이다.

- genre(str) : 웹툰 장르
- rating(float) : 웹툰 평점
- age(str) : 구독 제한 나이
- free(boolean) : 웹툰 무료 여부
- previous(int) : 작가의 전 작품의 개수(해당 작품이 나올때를 기준으로 함)
- episode(int) : 웹툰의 에피소드의 개수
- remake(int) : 원작 유무의 여부
- heart(int) : 웹툰이 받은 하트의 개수
- interest_num(target값)(int) : 웹툰의 구독자 수

데이터는 크게 범주형 데이터와 수치형 데이터가 존재한다. 범주형 데이터는 관측결과가 항목의 형태로 나타나는 데이터이고, 수치형 데이터는 관측결과가 수치로 나타나는 데이터이다.

범주형 데이터는 숫자로 표현되는 경우도 있지만 대부분 문자로 표현되어있다. 예를 들어 남자와 여자를 표현할 때 0과 1로 표현되어 있을 수도 있지만 대부분 '남자', '여자'라는 문자로 표현되어 있다. 다만 컴퓨터는 숫자만을 인식한다. 그래서 컴퓨터에게 기계학습을 시키기 위해서는 문자를 숫자로 encoding하는 processing 과정이 필요하다.

범주형 데이터를 processing하기 위해 label encoding과 one-hot encoding을 사용하려고 한다. label encoding은

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

<그림 6> label encoding 예시

항목에 label을 붙여주는 형식이다. 예를 들어 사과, 치킨, 브로콜리가 있을 때 사과에게 1, 치킨에게 2, 브로콜리에게 3이라는 label을 부여하는 것이다.

다만 label encoding에도 문제점이 존재한다. label encoding은 1부터 오름차순으로 숫자를 주는 것이므로 기계가 숫자 값을 가중치로 잘못 인식 할 수도 있다. 이는 예측 성능의 저하를 일으킬 수도 있다. 이를 해결하기 위해 숫자 값을 가중치로 인식하는 선형회귀 알고리즘은 지양하고 트리계열의 알고리즘을 사용하는 것이 좋다.[2)]

범주형 데이터를 processing하는 다른 방법으로 one-hot encoding이 있다. one-hot encoding은 항목별로 column을 만든다. 해당하는 항목의 column에 1을 주고 나머지 항목 column에는 0을 준다. 예를 들어 사과, 치킨, 브로콜리가 있을 때 사과 column, 치킨 column, 브로콜리 column을 만든다. 만약 사과라면 사과 column에 1을 주고 나머지 치킨과 브로콜리 column에는 0을 준다.

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

<그림 7> one-hot encoding

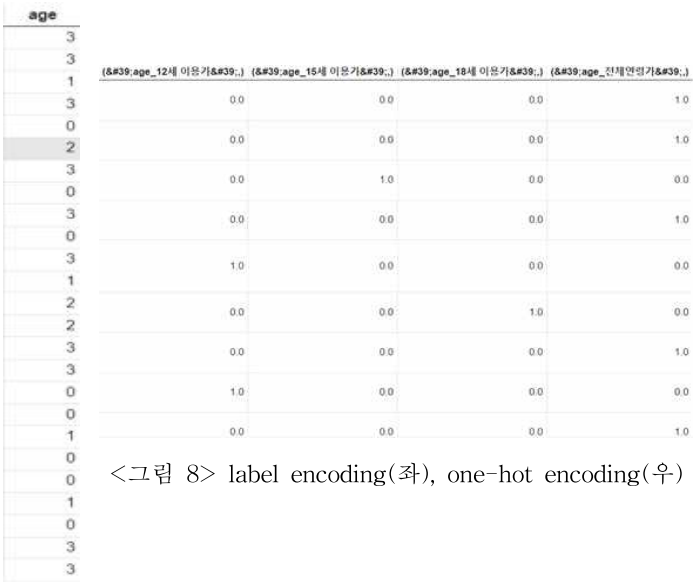
one-hot encoding도 만찬가지로 문제점이 존재한다. feature들 간에 강한 상관관계가 나타나는 다중공선성의 문제가 발생할 수도 있다. 이를 해결하기 위해서는 상관관계가 높은 열을 제거하던지 PCA[1)]를 사용해야 한다.

1) 기존 데이터의 분포를 최대한 보존하면서 고차원 공간의 데이터들을 저차원 공간으로 변환하는 기법

현재 데이터에서 범주형 encoding을 해야하는 column들은 이것이다.

- genre(str) : 웹툰 장르
- age(age) : 구독 제한 나이
- free(boolean) : 웹툰 무료 여부

앞의 column들을 scikit-learn의 LabelEncoder class와 OneHotEncoder class를 사용하여 범주형 processing을 구현하였다.



<그림 8> label encoding(좌), one-hot encoding(우)

수치형 데이터는 숫자로 되어있기 때문에 그대로 둬도 되지만 각 feature의 범위가 다르다면 scale을 하는 것이 좋다. 예를 들어 야구선수의 연봉을 생각해보자 feture는 나이와 월별 소득이 있다. 나이보다 월별소득이 훨씬 크기 때문에 월별소득에 치중한 학습을 하게 된다. 그러므로 범위의 왜곡을 막는 scale이 필요하다.

관측개체 번호	나이 (x_1)	월별 소득 (x_2)
1	30	3,620,000
2	13	0
3	21	600,000
4	61	500,000
5	7	0
⋮	⋮	⋮

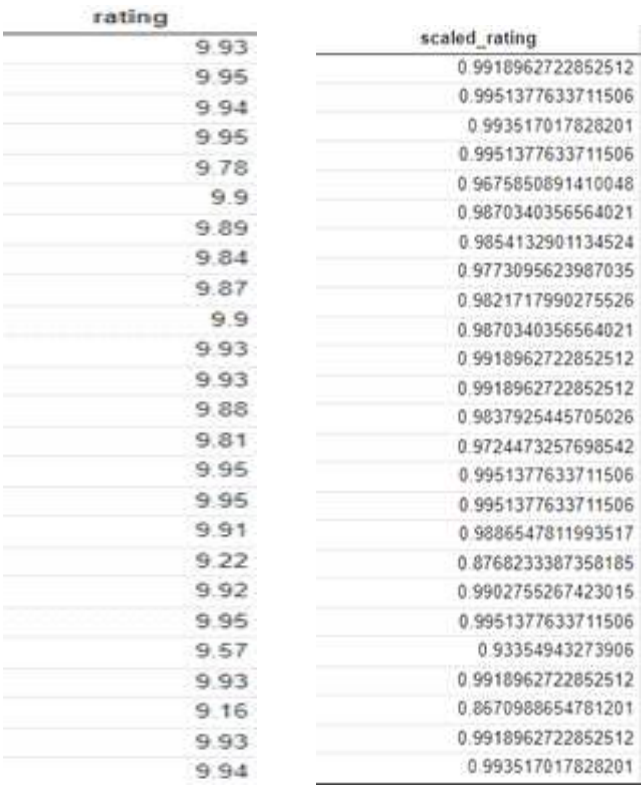
<그림 9> scale 예시

scale에 대한 연구가 활발하게 진행되고 있어서 다양한 scale 방법이 있지만, 필자는 Min-Max Scaling을 사용하였다. Min-Max Scaling이란 값의 범위를 0~1로 제한한 scale 방법이다.

현재 데이터에서 수치형 encoding을 해야하는 column들은 이것이다.

- rating(float) : 웹툰 평점
- episode(int) : 구독 제한 나이
- heart(int) : 웹툰 무료 여부

앞의 column들을 scikit-learn의 MinMaxScaler class를 사용하여 구현하였다.



<그림 10> 원본 데이터, MinMax scaling(우)

2.3 EDA(상관분석 결과가 너무 많기 때문에 따로 상관분석 결과를 pdf파일로 만들었습니다)

앞에서 data processing을 통하여 모든 데이터를 숫자로 바꾼 다음 상관분석을 해보려고 하였다. 다만 processing방법이 다양하기 때문에 다양한 상관분석 결과가 나왔다.

수치형 feature은 Min-Max Scaling을 고정으로 하였다. 어떠한 Scaling을 하여도 동일한 상관 분석 수치를 반환하기 때문이다. 다만 범주형 feature는 모든 경우의 수로 label encoding과 one-hot encoding을 적용시켰다. 왜냐하면 genre, age, free 모두 label encoding을 적용시키는 것과 genre에는 label encoding, age와 free에는 one-hot encoding을 적용하고 상관 분석을 하는 것이 다른 결과를 보여주기 때문이다.

heart(수치형)와 episode(수치형)만 약 |0.5| 정도의 수치를 보이며 적당한 상관관계를 보여주었다. 하지만 나머지는 어떠한 processing 방법을 사용하더라도 |0.2|보다 작은 수치로 상관관계가 상당히 낮았다.

앞의 상관 분석을 통하여 상관관계가 |0.1|보다 작은 genre, age, remake column은 제거하였다. 제거하고 남은 column들은 이것이다.

- rating(수치형) : 웹툰 평점
- free(범주형) : 웹툰 무료 여부
- previous(수치형) : 작가의 전 작품의 개수(해당 작품이 나올 때를 기준으로 함)
- episode(수치형) : 웹툰의 에피소드의 개수
- heart(수치형) : 웹툰이 받은 하트의 개수
- interest_num(target값) : 웹툰의 구독자 수

2.4 Data modeling

앞에서 Data processing과 EDA를 통하여 기계학습을 준비를 마쳤다. model은 선형회귀 알고리즘 3개(LinearRegression, Ridge, Lasso)와 tree알고리즘 3개(RandomForest, XgbBoost, LightGbm)를 사용할 것이다. 학습된 모델의 정확도를 측정하기 위해 MAE를 사용할 것이다.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|$$

<그림 11> MAE 공식

앞의 EDA에서와 같이 수치형 feature은 Min-Max Scaling을 고정으로 하였지만, 범주형 feature은 모든 경우의 수로 label encoding과 one-hot encoding을 적용시켰다. free에 label encoding을 적용하고 modeling하는 것과 free에 one-hot encoding을 적용하고 modeling 하는 것이 다른 결과를 보여주기 때문이다. 다만 앞의 EDA를 통해 범주형 column은 free밖에 안 남아서 많은 경우의 수가 나오지는 않는다.

	free(labeling)	
선형회귀	LinearRegression	286.00826730651556
	Ridge	287.9927528356752
	Lasso	286.0099421126427
tree계열	RandomForest	275.42249336975834
	XgbBoost	283.4568894234241
	LightGbm	279.77083261768075

<표 1> free(labeling) 일 때 MAE

	free(one-hot)	
선형회귀	LinearRegression	286.70842928490055
	Ridge	287.2959373050697
	Lasso	286.70397547906924
tree계열	RandomForest	279.3119914005591
	XgbBoost	284.22984192029526
	LightGbm	279.24827721816786

<표 2> free(one-hot) 일 때 MAE

앞의 modeling결과를 보면 free(labeling)일 때 tree model인 LandomForest를 사용한 결과가 제일 좋다. 이 때 MAE는 275.422493369758334이다. 평균적으로 구독자 수를 예측할 때 275명의 오차가 발생한다는 의미이다.

또한 **2.2**절에서 선형회귀는 labeling encoding할 때 숫자를 가중치로 인식하여 tree model이 더 좋다고 했다. 결과는 기대했던 데로 tree계열 model이 더 좋게 나왔다.

3. 한계

본 프로젝트는 주어진 문제가 아닌, 실제로 필자가 관심있는 문제를 풀어보았다. 그래서 직접 data collection, data processing, EDA, data modeling을 총괄하였다. 일반적인 대회에서는 좋은 데이터를 제공하기 때문에 data collection단계를 생략할 수 있었지만, 현실 문제를 해결해야 하는 것이다 보니 모든 단계를 직접 수행해야 했다. 기계학습을 하는데 모든 과정을 수행해 본다는 점에서 얻는 것이 많았다. 하지만 3개월이라는 짧은 시간동안 모든 부분을 집중할 수가 없었다는 아쉬움이 남는다. 특히 data modeling 단계에서 많은 paper를 읽어보고 다양한 model을 사용해보고 싶었는데, model에 대한 특징도 잘 모르고 사용만 한 것에서 아쉬웠다.

또한 본 프로젝트에서는 data collection을 직접하여 현실 데이터를 모아서 modeling을 하였는데 생각보다 성능이 안 나왔다. 대회에서 제공하는 데이터와 현실 데이터는 엄청난 차이가 있다는 것을 깨닫는 시간이 되었다. 현실의 문제를 해결하는 data scientist가 되기에는 한참 부족하다는 것을 느꼈다.

4. 참고문헌

1) https://www.hankyung.com/international/article/
202208227609i