**Machine Learning & Data Mining**

**Team Project**

201421074 백노연

201520940 윤선용

201520945 고윤혁

201521149 정유석

201723271 김황대

1. **Title (주제)**

나도 성공할래 : Application의 주요 정보를 기반으로 평점 예측

1. **Business Understanding**
2. **모델의 주 목적**

Application의 주요 정보를 기반으로 이 앱이 평점 몇점을 받을 수 있는 앱인지 예측하여 준다. 평점이 높다는 것은 사용자들이 이용하기 편하고, 앱이 시장성 있다는 것을 의미한다. 개발자가 앱을 개발하여 마켓에 올리기 전에, 자신의 앱에 해당 되는 model의 fecture들에 넣어 성공률을 예측해볼 수 있다면, 이미 성공한 어플에 비해서 현재 개발한 앱의 부족한 점이 무엇인지, 현재 앱이 사용자들이 원하는 needs를 잘 수용하고 있는지 파악 할 수 있을 것이다. 뿐만 아니라 앱이 만들어지기전 앱의 Feature나 description 을 나도 성공할래 모델에 넣고 얼마나 평점을 받을지 예측해서 앱의 특징이나 앱에 대한 설명을 보완하여 완성도 높은 앱을 개발하도록 도와주는게 이 모델의 목적이다.

1. **모델의 실용적 측면**
2. **앱의 여러가지 정보에 의한 예측**

앱을 개발하기 전 기획단계에서 앱이 유저들에게 어떤 호응을 이끌어낼지에 대해 앱의 주요 피쳐들을 평점을 예측해볼 수 있다. 앱을 마켓에 출시하기전에 개발한 앱의 feature들을 이 모델에 넣고서 과연 이 앱이 평점 몇점을 받을지에 대한 예측을 하는데 사용해 볼 수 있다. 그리고 예측값이 낮다면 어떤 feature들이 성공한 앱에 비해 낮은 값을 가지고 있는지 확인하여 앱의 단점을 보완하도록 알려준다.

1. **앱 설명에 의한 예측**

A에서 활용할 수 있는 정보를 해석하여 보더라도 “그래서 그 앱이 무슨 앱인데?” 라는 질문에 대답해 주기는 어렵다. A에서 활용하는 정보에 그 앱에 대한 설명이 들어있지 않기 때문이다. 하지만 앱의 설명 부분을 끌고와서 그것으로 평점을 예측할 수 있다면 A와 같이 앱에 대한 평점 예측을 해보면서도 어떤 컨셉의 앱이기 때문에 그러한 결과를 냈다는 해석이 가능하므로 더 가치있는 모델이 될 수 있을 것이다. 이러한 해석에 기반을 둔다면 앱에 대한 설명이 앱의 기능을 대변한다고 볼 수 있고 유저들이 높은 평점을 준 앱은 유저들의 니즈를 충분히 반영한 가치있는 앱이라고 볼 수 있다. 그렇다면 앱의 핵심기능을 유저들에게 소개하는 글을 평가해주는 모델로서 사용할 수 있을것이다.

조금 실용성있는 접근을 해보면 앱의 기획 과정이나 등록시 소개 과정에서 미리 평점을 받아보고 이 앱이 과연 얼마나 쓸모있는 앱인지에 대한것을 알려주는 서비스를 생각해볼 수 있다.

1. **Data Understanding**

**사용한 데이터 셋 출처**: Kaggle (<https://www.kaggle.com/ramamet4/app-store-apple-data-set-10k-apps>)

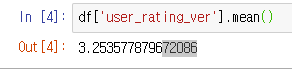
사용할 Dataset은 Kaggle의 Mobile App Store에서 다운로드하였으며 7197개의 sample과 17 개의 feature를 포함하고 있다. 이 중 사용될 주요 feature를 15개로 지정하였는데 해당 목록은 아래와 같다.

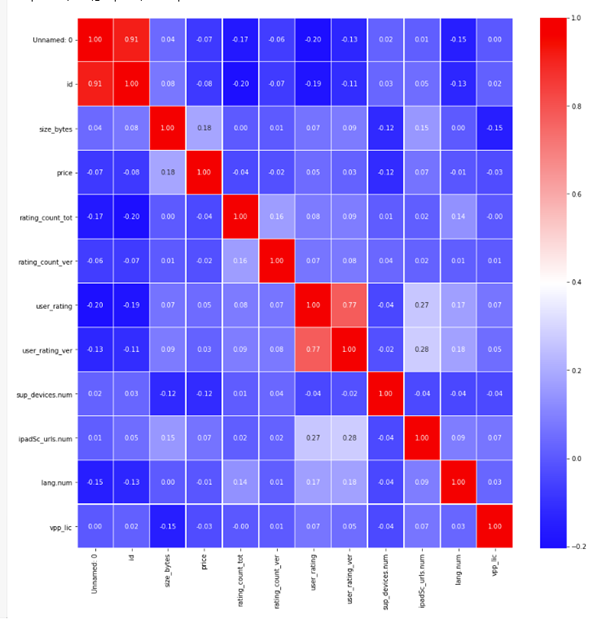
|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **번호** | **ID** | **번호** |
| TrackName: Application | 이름 | Count Rating | 연령제한 |
| Size Bytes | 앱 크기 | Prime Genre | 장르 |
| Currency | 통화 | Supported Devices Number | 호환가능 기기 수 |
| Price | 가격 | Screenshot Number | 보여주는 스크린샷 수 |
| Rating Count Total | 전체 평점 수 | Language Number | 지원 언어 |
| User Rating | 평균 평점 | Application Description | 앱 설명 |
| Version | 현재 버전 |  |  |

위와 같은 정보를 포함하고 있는 Dataset을 사용할 것이며 이 중 주(Target) 데이터는 UserRating 즉, 평점으로 선정하고 이를 예측하는 모델을 구축하는 것을 목표로 하였다.

**데이터 셋의 이해를 위해 몇 가지 궁금한 점에 대해 조사해보았다.**

**앱의 전체 평점은 몇점일까? 일부 피쳐들사이의 상관관계 분석**

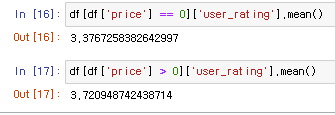
****상관관계 분석만으로는 의미 있는 상관관계를 찾기는 어려웠다

 **장르별 평균 평점은?**

.

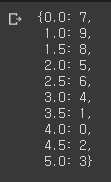
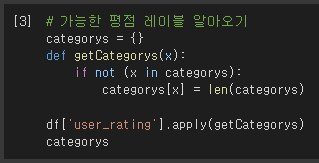
.

**무료앱과 유료앱의 평점차이?**

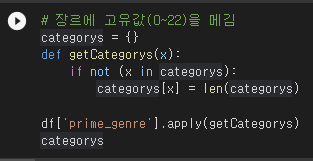
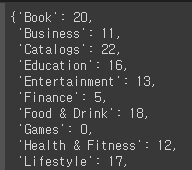
****

앱 스토어의 경우 유료앱이 평균적으로 더 좋은 평점을 받았다.

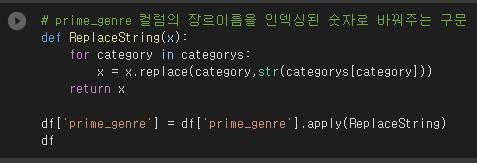
1. **Data Preperation**
2. **평점 레이블 알아오기 : 어떤 평점들이 있는지 확인한다.**



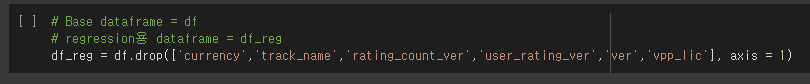
1. **장르 인덱싱 : 어떤 장르들이 있는지 확인 후 번호를 부여한다. (지면상 일부생략)**

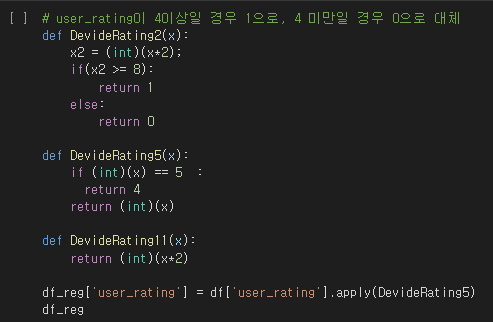
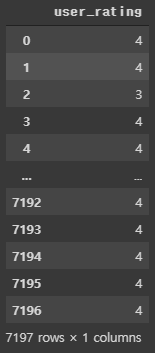
1. **장르 컬럼을 숫자로 변환**



1. **불필요한 컬럼 드랍 : Model마다 다르게 드랍한다.**



1. **User Rating 구간 분할 : classfication을 위해 적절하게 구간단위로 나눠주었다.**

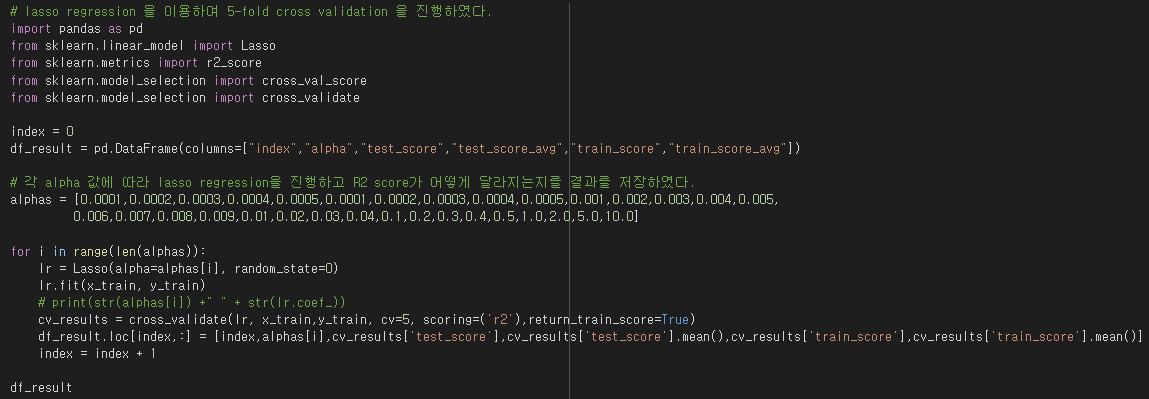
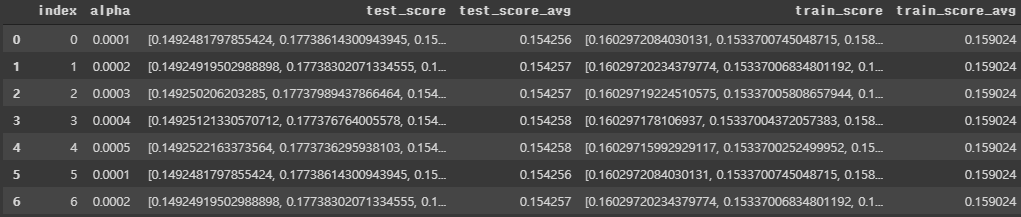
 

1. **Cont\_rating의 4+ 와 같은 스트링을 +를 제거하고 문자열로 변환 :**

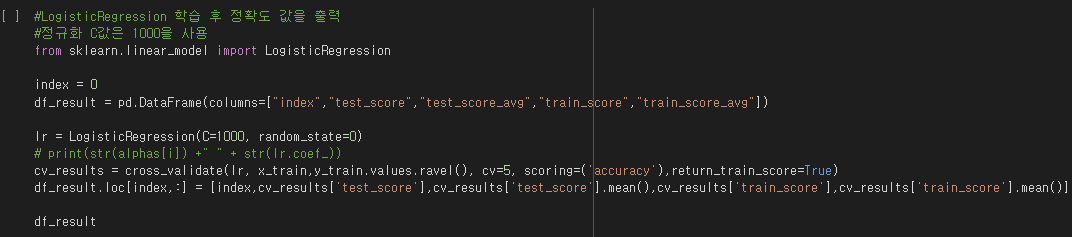


**5-1. Modeling & Evalution : Regression**

1. **DivideRating 5 : 다섯구간으로 나눈 결과 :** 가장 먼저 평점을 다섯 구간으로 나누어서 테스트를 진행하였다. (구간분할에 관해서는 4. Data Preparation 참고)
2. **LinearRegression (lasso) :** Lasso 모델을 사용하여(5-CV) 여러 alpha 값에 따른 정규화 정도에 따라 R2 score를 측정하였다. 전체적으로 Training set에서 0.154 validation set에서 0.159의 R2 score를 보여주었으며 alpha값이 극단적으로 커지는 경우를 제외하면 score의 차이는 그리 크지 않았으며 0.15%의 낮은 R2 score를 보였다. Alpha가 과도하게 커질수록 너무 많은 정보의 손실이 일어나 정확도가 급격히 하락하는 모습을 볼 수 있었다. 최종적으로 테스트 셋에서 0.135948의 R2 score를 보였다.

1. **Logistic regression :** 다음으로 Logistic regression 방법을 사용하였는데 5Fold-CrossValidation방법을 사용하였다. 트레이닝 셋에서 Accuracy는 0.66 이 나왔으며 트레이닝 셋에서 R2-Score는 -0.33이 나왔다. 테스트 셋에서 최종 실행결과 0.66의 Accuracy와 -0.35의 R2\_Score를 보였다.

1. **DivideRating 2 : 두 구간으로 나눈 결과**
2. **LinearRegression (lasso) :** 가장 결과가 좋은 0.01의 alpha 사용시 0.157394의 R2 Score를 보여주었다.



1. **Logistic regression :** Logistic regression 테스트 방법으로 5Fold-CrossValidation방법을 사용하였다. 트레이닝 셋에서 Accuracy는 0.66이, R2-Score는 -0.33이 나왔다. 최종적으로 테스트 셋에서 0.67의 Accuracy와 -0.33의 R2\_Score를 보였다.
2. **DivideRating11 : 11 구간으로 나눈 결과**
3. **Linear Regression :** 0.01의 alpha 사용시 0.124740의 R2Score를 보여주었다.



1. **Logstic Regression :** 트레이닝 셋에서 -0.40의 R2-Score를 보여주었으며 트레이닝 셋에서 0.36의 Accuracy를 보여주었다. 최종적으로 테스트 셋에서 0.37의 Accuracy와 -0.44의 R2-score를 보여주었다.

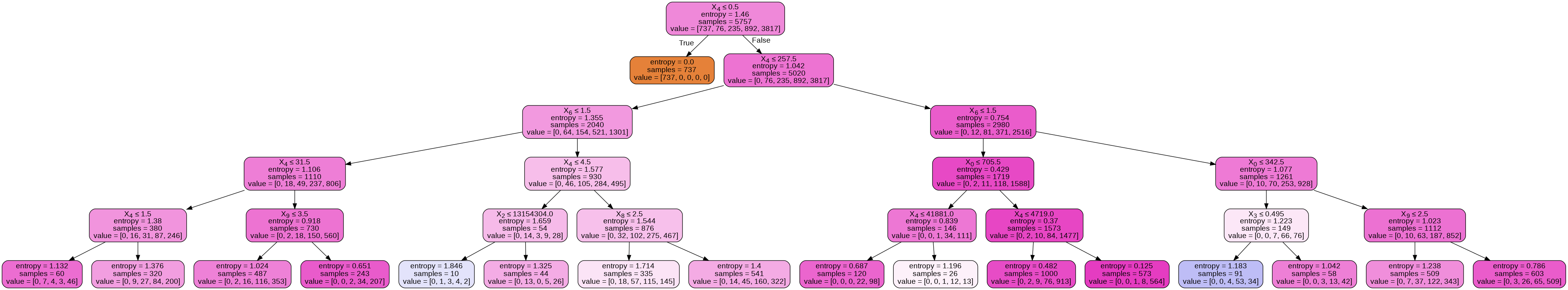
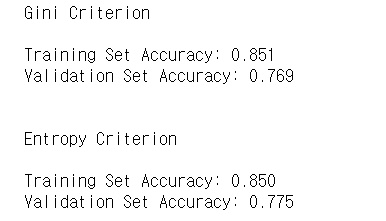
**결과에 대한 해석**

**구간에 따른 R2 Score / Accuracy 비교**

|  |  |  |  |
| --- | --- | --- | --- |
| **분류 / 구간** | **2구간** | **5구간** | **11구간** |
| **Linear Regression** | 0.157394 / - | 0.135948 / - | 0.124740 / - |
| **Logistic Regression** | -0.35 / 0.66 | -0.33 / 0.67 | -0.44 / 0.37 |

먼저 Linear Regression의 경우 구간에 변화에 따른 R2 Score 차이는 크지 않았지만 R-squared가 매우 낮게 나왔다. 그 이유를 해석해보자면 현재 주어진 피쳐들은 대부분이 명목척도(nominal scale)로서 변환된 값이 많이 있다. 그런데 Regression을 수월하게 하기 위해서는 등간척도(Interval scale)로서 둘 사이의 양적인 간격을 측정할 수 있어야 할 것으로 생각된다. 예를 들어 장르와 같이 Game = 1, News = 10으로 변환된 피쳐를 보자면 이 값은 단지 레이블을 나타내는 값일뿐 News 가 Game의 10배의 가치를 가짐을 의미하는 값은 아니다. 따라서 Regression을 하기에 적합하지 않은 피쳐들이 많았으며 상관계수 검사에서도 낮은 상관계수들이 관찰되었다. Logistic Regression의 경우 classification 문제를 푸는데 좋은 성능을 보이는 만큼 2구간과 5구간에서 높은 정확도를 보였지만 구간의 수를 11구간으로 넓히면 0.36으로 성능이 급격하게 감소하는 모습을 볼 수 있었다.

**5-2. Modeling & Evalution : Classification [Decision Tree]**



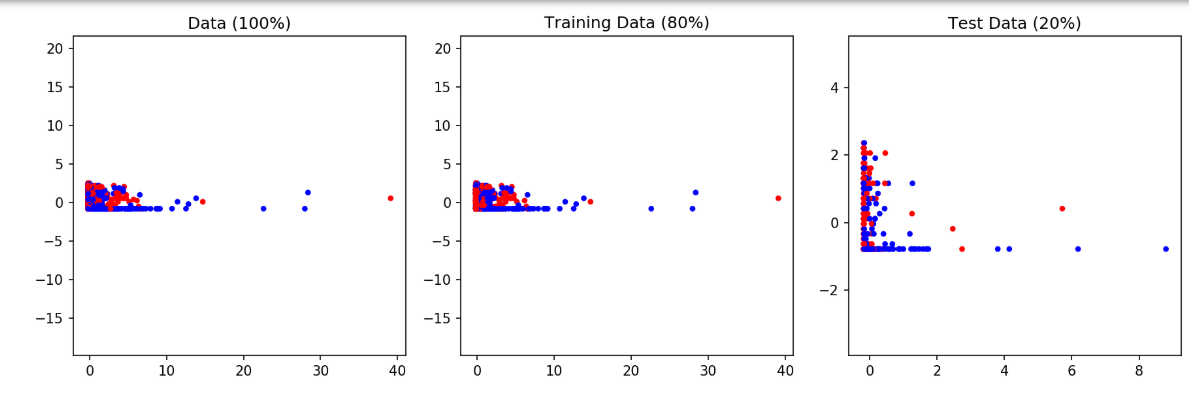
의사결정 트리를 사용하기 위해 전체 Data Set 에서 Target이되는 uesr\_rating을 다로 분리했고 구간을 5개로 나눴다. 또한 분류에 필요없는 feature들을 판단하여 제거하였고 문자로된 feature을 넘버링 해줬다.

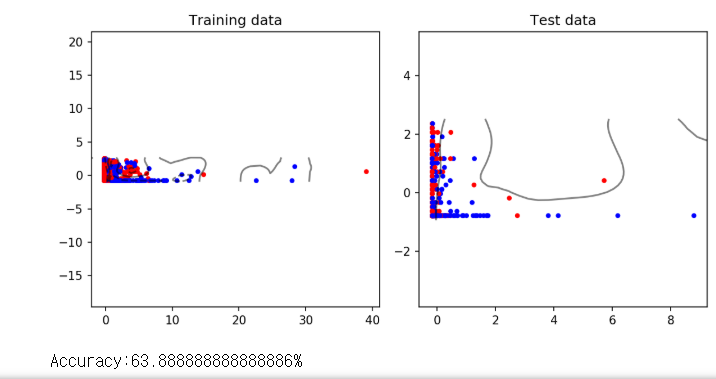
의사결정 트리의 하이퍼파라미터인 엔트로피와 지니계수를 비교하여 적절한 하이퍼 파라미터를 찾아줬다. 또한 depth 를 결정하기위해 3~10까지의 depth를 확인해봤으며 가장 결과가 좋은 depth를 결정해줬다.

엔트로피와 지니계수의 차이는 엔트로피가 좀더 좋은 성능을 낼것이라 예상했으나, 0.001 차이로 큰차이가 없었고 depth의 따른 차이는 엔트로피나 지니계수의 차이보다 조금 더 두드러지게 보였으나 역시 큰차이가 없었다.

결과적으로 최적의 하이퍼파라미터를 선택하여 validation set 에대한 정확도를 77%의 얻어냈으며 분류할수있는 feature의 수가 많기 때문에 위와같은 높은 정확도가 나왔을거라고 추측한다.

**5-3. Modeling & Evalution : SVM**





SVM 을 사용하기 위해서는 결과에 영향이 높은 feature를 찾는것이 먼저였다.

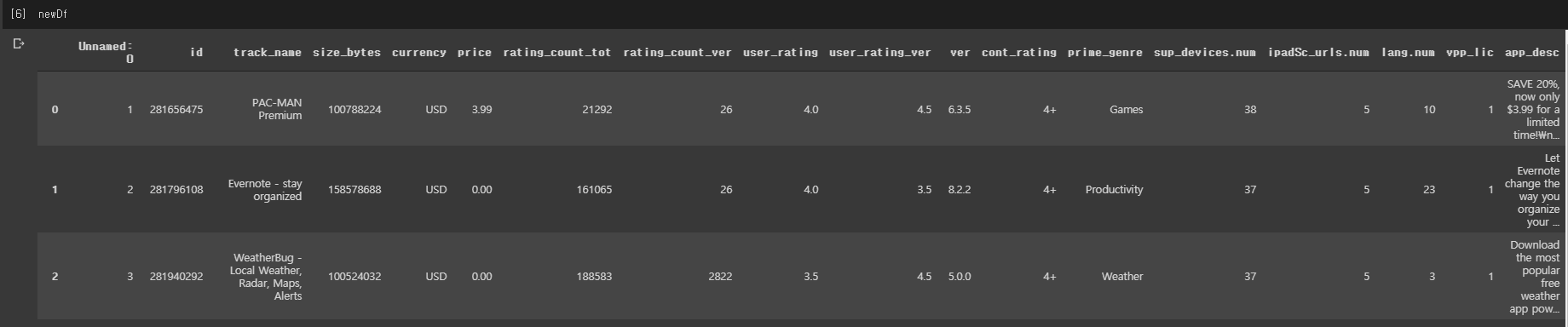
분류에 필요한 feature가 이 data set에서는 14개라고 판단했고 만약 feature를 전부 svm 모델에 넣어서 분류해서 EDA하려면 14차원의 EDA가 될것이기에 Feature를 선별하는 작업이 필요했다. 이를 위해 브루트 포스 방식을 이용해 분류에 가장 적합한 두개의 feature를 찾았다. 그리고 평점이 4이하 인 구간과 4이상인 구간으로 나누어 EDA를 했을 때, 보기 좋도록 데이터를 전처리했다. 또한 Data의 sample들이 너무 광범위하게 퍼져있기에 정규화의 과정도 거쳤다.

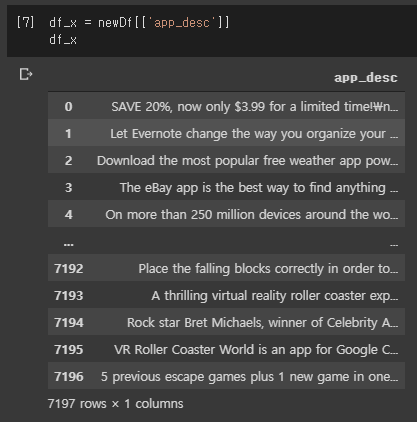
그 결과가 위의 사진이다. 파랑색 점이 평점이 4이상인 앱이고 빨간색점이 4이하의 앱들이다 사진에 보이는것과 같이 SVM으로 데이터를 학습한 모델이 어느정도로 파랑색 점들을 분류한것을 볼수있다. 정확도는 63%가 나왔다.

하지만 위 SVM 모델은 너무 많은 Data를 버리고 target의 구간도 2개로 밖에 나누지 못했기 떄문에 우리가 목표한 모델에는 적합하지 못했다.

**5-4. Modeling & Evalution : Text classification with RNN**

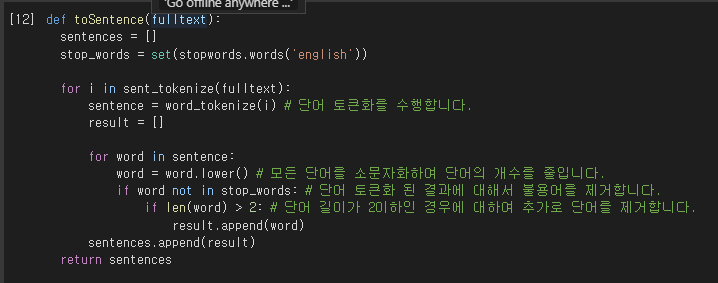
AppStore데이터와 해당 앱의 설명정보를 병합하면 다음과 같다.



(왼쪽그림) 그 중 app\_desc컬럼만을 추출하였다. 예를 들어 0번 샘플 “PAC-MAN Premium” 의 description을 출력해보면 다음과 같다. 'SAVE 20%, now only $3.99 for a limited time!\n\nOne of the most popular video games in… (생략)

Y 데이터로는 5구간으로 분류된 평점 데이터를 사용하였다. 0~1, 2~3, 4~5, 6~7, 8~10.

트레이닝 셋과 테스트 셋을 0.9:0.1의 비율로 분할하였다.



다음으로 Desciption을 단어 단위로 쪼개는 함수를 작성하였는데 (위쪽그림) 이 함수는 먼저 전체 스트링(Desciption전부)을 sent\_tokenize를 통하여 문장단위로 쪼개고 각 문장을 다시 word\_tokenize로 단어 단위로 쪼갠다. 그런데 여기에서 금지단어와 길이가 2이하인 단어는 포함하지 않도록 하였다.

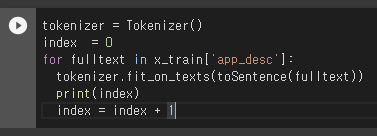
예를들어 1941번 샘플에 대한 실행 예시는 다음과 같다.

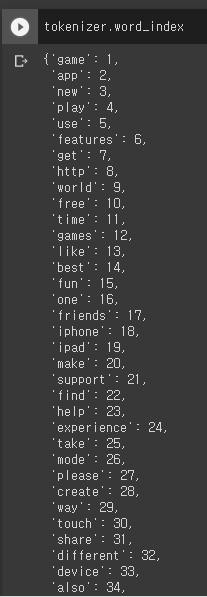
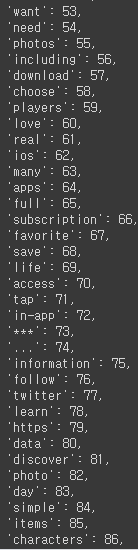
FullText = 'This is a stereogram application.\n\nIn general, stereogram uses objects and scenery as the theme.\nHowever, the theme of this… (생략),

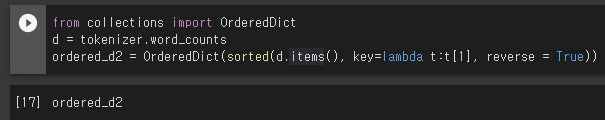


위에서 설명한 toSentence함수를 거치면 (위쪽 그림) 아래와 같이 변환된다.

[['stereogram', 'application'],['general', 'stereogram', 'uses','objects', 'scenery', 'theme'],['however', 'theme',… (생략)

다음으로는 Tokenizer를 사용하여 x\_train의 app\_desc에 대해 toSentence로 변환한 모든 시퀀스 값에 대해 fit\_on\_texts를 실행하였다. 이 작업을 실행하면 tokenizer에서 각 단어에 대해 번호를 붙혀주게 된다. 실행하면 아래와 같이 단어별로 번호가 붙는다. 이 작업은 새로운 단어가 들어올때마다 dictionary를 참고해 단어가 이미 등록되어있는지 아닌지에 따라 없다면 번호를 붙혀주게 되므로 전체 description에 대해 실행하면 꽤 많은 시간이 소요되게 된다. (7000개의 테스트셋의 모든 설명에 대해 10분 가량 소요되었다.)



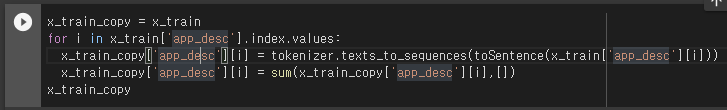
실행결과 각 단어에 번호가 붙은 모습을 확인할 수 있다. (왼쪽그림)

또한 tokenize에서는 각 단어가 몇 번 쓰였는지도 기록하여준다. (위쪽 그림, word\_counts) (Dictionary에 기존 단어가 없다면 0으로 추가, 있다면 숫자+1.)

이 정보를 이용해서 단어가 많이 쓰인 순으로 정렬해보았으며 그 결과는 아래와 같다.

OrderedDict([('game', 8873),('app', 8524),('new', 5599),('play', 5020),('use', 3920),('features', 3723),'get', 3588),...생략

X\_train 데이터 셋에서 1000번 이상 사용된 단어들만 추린 결과인데 ‘game’이 8873번으로 1위를 기록하였고 ‘app’이 8524번으로 2위를, ‘new’가 5599번으로 3위를 기록하였다. ‘http’가 3367번 쓰인점은 흥미로웠는데 홈페이지 소개에 대한 문구가 많이 들어가는 것으로 추측할 수 있겠다. 또한 game 단어가 여러 번 등장하는 것으로 보아서(‘games’ 또한 3063번이나 사용되었다.) 게임 앱이 많음을 유추가능하다.



다음으로는 위에서 각 단어에 붙혔던 번호에 따라서 각 단어를 번호로 바꿔주는 작업을 진행하였다. Tokenizer.text\_to\_sequences를 실행하면 해당 작업을 진행해준다.

이 작업을 완료한 예시로 위에서 실행했던 1941번 샘플의 예를 들어 비교해보면 결과는 아래와 같다.

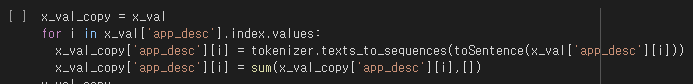
|  |  |
| --- | --- |
| [['stereogram', 'application'],['general', 'stereogram', 'uses','objects', 'scenery', 'theme'],['however', 'theme',…(생략) | [9501,187,1723,9501,570,547,  3559,1050,853,1050, … (생략) |

앞의 두개중 187번을 위의 tokenizer.word\_index 에서 찾아보면 application과 같음을 확인할 수 있다.

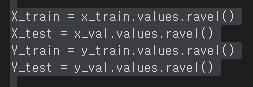
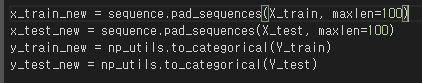
'questions': 186,'application': 187, 'itunes': 188,

같은 방법으로 테스트 셋(x\_val로 명명되어있지만 test set이다) 에 대해서도 fit\_on\_text, texts\_to\_sequences를 실행하였다.



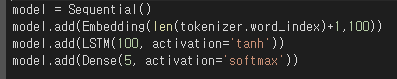


**RNN 적용**

 (1) (2)

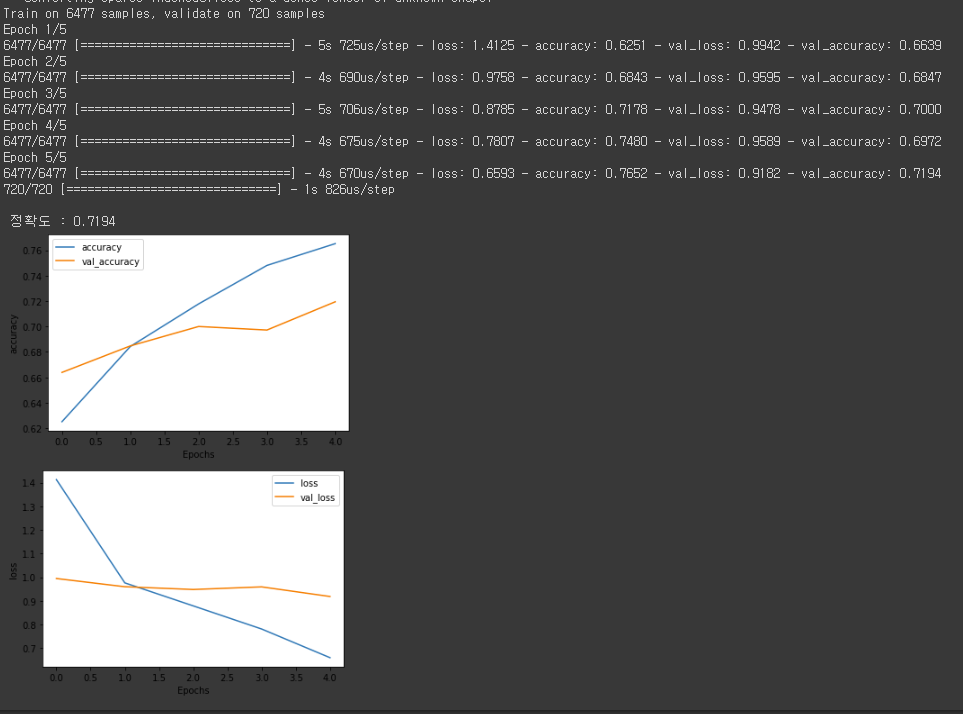
먼저 위에서 전처리한 x\_train과 x\_val(이름은 val이지만 test셋이다. 기존의 테스트중 속도를 줄이기위해 val로 작게 쪼갠 데이터를 사용하였다.) 그리고 각각의 y데이터 셋에 대해 ravel을 사용하여 1차원 데이터로 펼쳐주었다. (그림1)

다음으로 100개의 단어가 안되는 샘플에 대해 100개까지 0값으로 채워주는 pad\_sequence작업을 수행하였다. (그림 2)

(3)

모델은 Embedding + LSTM + Dence(softmax)로 구성하였다. (그림 3)

실행결과는 아래와 같다.



지면관계상 생략하였지만 epochs = 10 에서는 0.50 정도의 정확도를 보여주었으며 최종적으로 여러 번 시도 끝에 batchsize 512, epochs = 5이 0.71로 가장 좋은 결과를 보여주었다.

**결과에 대한 해석**

RNN의 경우 5구간으로 분류하였음에도 불구하고 70%의 꽤 높은 정확도를 얻을 수 있었고 기존의 데이터에서는 높은 정확도가 나오더라도 그 앱이 무슨 앱인지에 대한 정보를 포함하고 있지 않았기때문에 쓸모있는 앱으로서 높은 평점을 얻은 것이라고 해석하기는 힘들었지만 RNN을 사용한 분류의 경우 해당 앱이 무엇을 하는 앱인지에 대한 정보가 description에 포함되어 있으므로 이는 소비자인 유저들의 니즈를 잘 충족시키는 앱이 높은 평점을 받은 것으로도 해석이 가능할 것이다. 또한 너무 많은 epochs는 정확도를 오히려 떨어뜨리는 모습을 관찰할 수 있었다. 또한 시간상의 문제로 비록 시도하지는 못했지만 CNN모델을 결합하는 방법도 고려해볼 수 있을 것 같고 높은 평점을 받은 앱에서 자주 나타난 빈도수의 단어들을 골라본다면 이러이러한 단어가 나타난 앱의 평점이 높았다라는 정보도 얻을 수 있을 것이라고 생각된다.

1. **결론**

app을 예측하기위해서 위의 4가지 회귀, 분류 모델을 사용해보았다. 어떤 모델이 무조건적으로 좋다고는 할 수 없지만 높은 정확도를 나타낸 모델이 있었다.

 가장 높은 정확도를 나타내는것은 의사결정트리였다. 이러한 결과는 의사결정트리는 피쳐마다 구간을 나눠 분류를 하는데, 분류를 할수 있는 피처의 수가 14가지가 된다. 구간이 많이 눠진다는 것은 분류를 조금더 구체적으로 할수있다는 것이고, 엔트로피나 지니 계수에따라 분류를 하니 가장 직관적이면서 높은 예측도가 나온것이라고 생각된다.

하지만, 위 data set으로만 앱의 평점을 예측하기에는 부족하다고 할 수 있을 것이다. 왜냐하면 위 data set에는 가장 중요하다고 생각되는 ‘해당 app이 무엇을 하는 앱인지’에 대한 정보가 빠져있기 때문이다.

 따라서 app의 설명으로 부터 평점을 예측한 rnn모델이 가장 가치있다고 생각되어 진다. app 디스크립션에는 해당 앱의 특징들과 장점 들이 모두 들어가 있고, 이는 app이 무엇을 하는 app인지를 대변한다고 할수있기 때문이다. 그렇지만 이 말이 data set을 가지고 예측했던 모델이 가치없다는 뜻은 아니다. 그 모델 또한 data set에 근거하여 높은 평점을 가지기 위한 정보들이 들어있기 때문이다.

 결론적으로 앱의 App의 평점을 예측하는데에는 data set과 설명 데이터를 전부 활용할 수 있어야 할 것이며 의사결정트리와 RNN모델을 전부 사용해 볼 수 있을 것이다. 또한 개선할 점이 있다면 비록 시간상의문제와 난이도의 문제로 해보지는 못하였지만 장르별로 앱의 설명을 분석한 다음 해당 앱의 장르에서 높은 평점에 기여하는 feature들을 찾아낸다면 좀더 구체적이고 정확한 모델이 될 수 있었을 것이라고 생각한다.