Crowd funding analysis

2017103972 김성연
2016104122 박기범
2015104175 박우진
2014104162 홍성현

Index

Bigdata project - Crowd funding analysis

1st

Motivation

Why we chose this topic

2nd

Requirement & background

For this project, we have to know these points

3rd

Data acquisition

How to getter and pre-process law data

4th

Data storage

The storage what we use for this project

5th

Data analysis

How to analysis our law data

6th

Data visualization

How to visualize our analysis result

7th

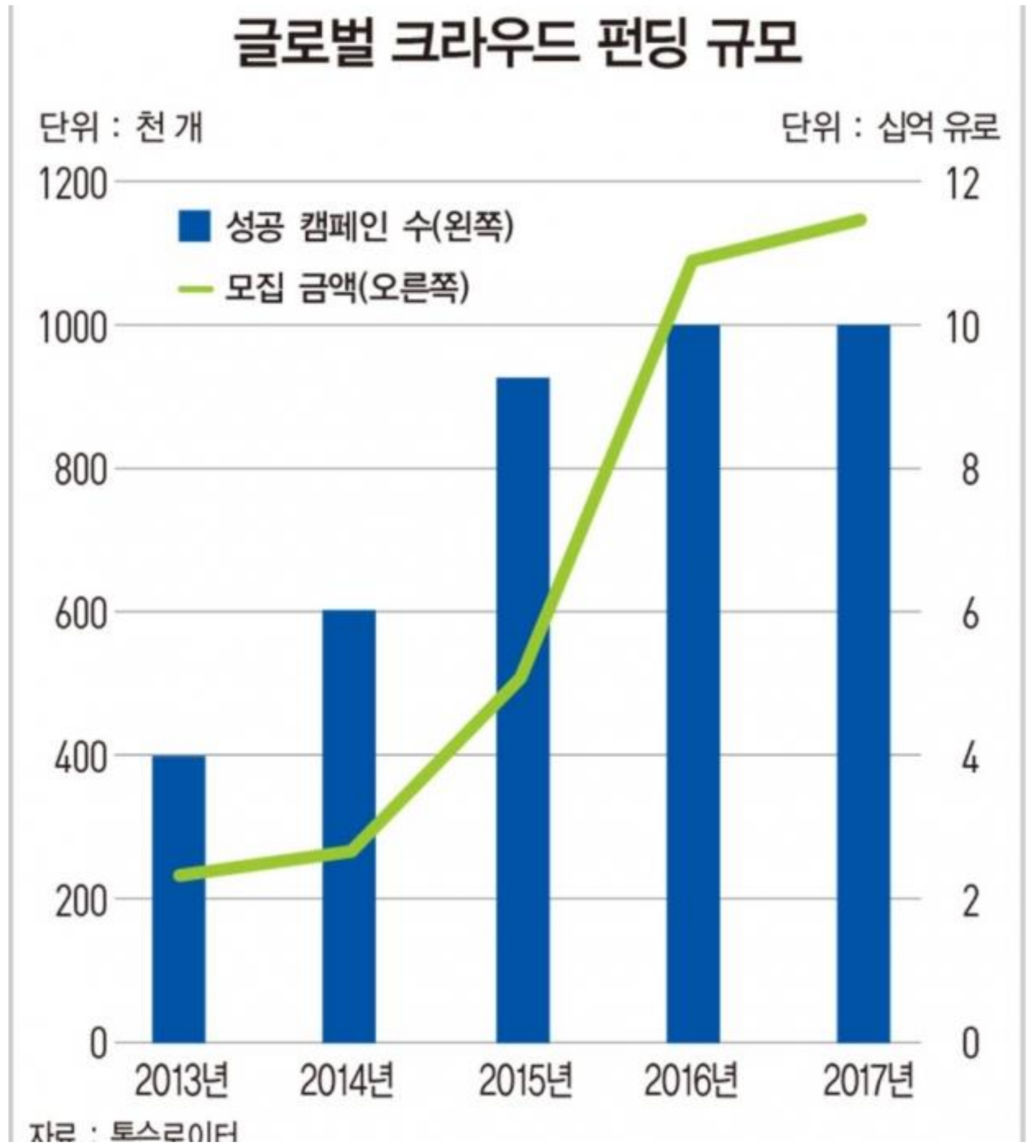
Result & individual Role

Conclusion of our project and introduce individual Role

FINISH

Motivation

Bigdata project - Crowd funding analysis



Crowd funding is the new trend.

And, still growing!

But, there are some problems..

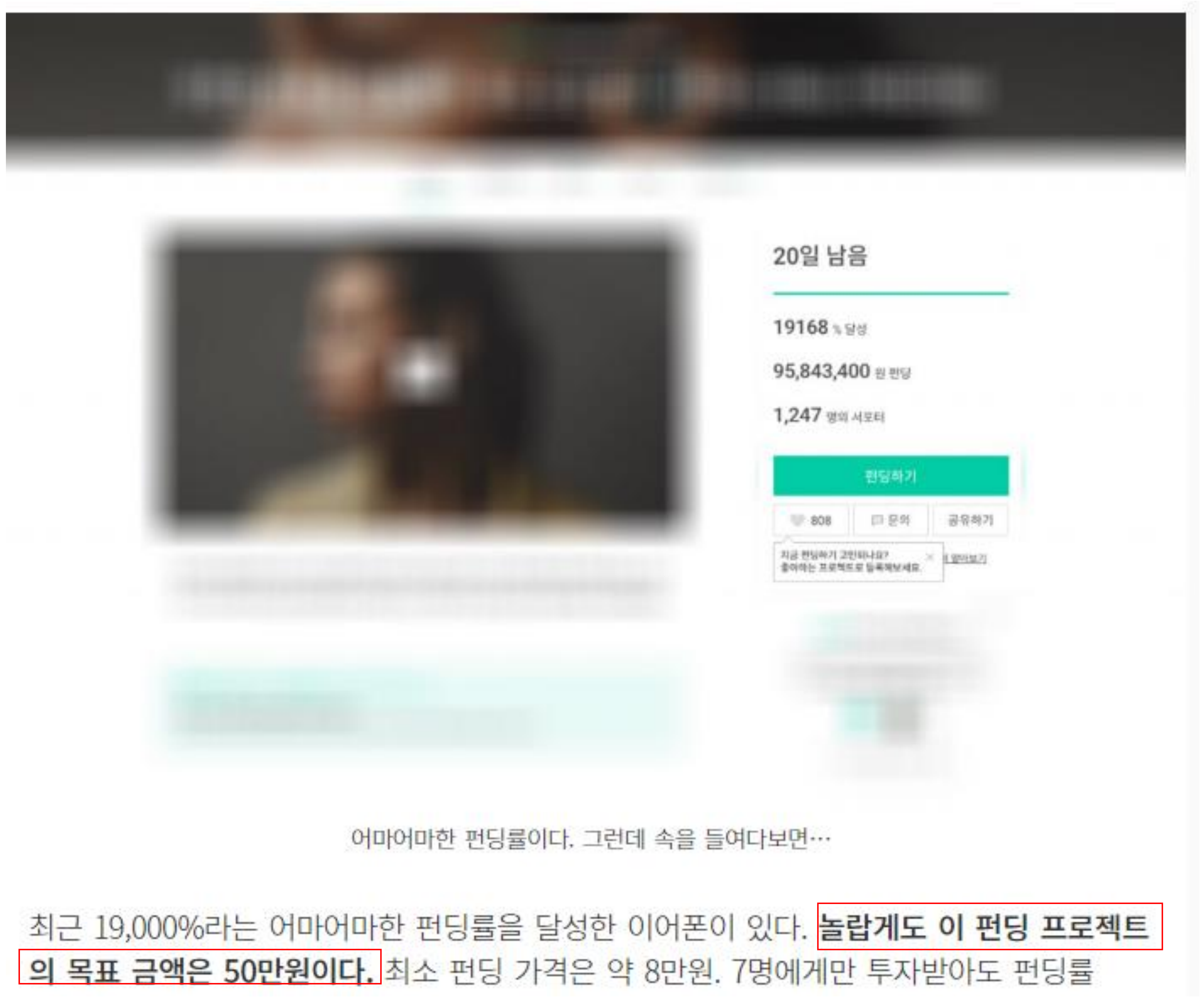
Motivation

품질 미달의 대표 주자는 단연 '퀸메이드 400W 앱솔루트 무선 청소기'다. 메이저 브랜드 무선 청소기의 출력이 보통 150W다. 그런데 퀸메이드는 400W 출력을 내세웠다. 그야말로 혁신 아닌가. 하지만 이 제품은 아예 사용할 수 없었다. 전원조차 켜지지 않는 탓에 [얼리 펀딩] 기사로도 싣지 못했다. 불량을 호소하는 투자자는 한둘이 아니었다. 결국, 문제가 발생한 제품에 대해 전량 환급 조치하는 초유의 사태가 벌어졌다. 이외에도 품질 관련 CS가 줄을 이었고, 크라우드펀딩 페이지는 난장판이 되었다.

1. Poor quality

There are some bad product

But it is not easy to refund..



어마어마한 펀딩률이다. 그런데 속을 들여다보면…

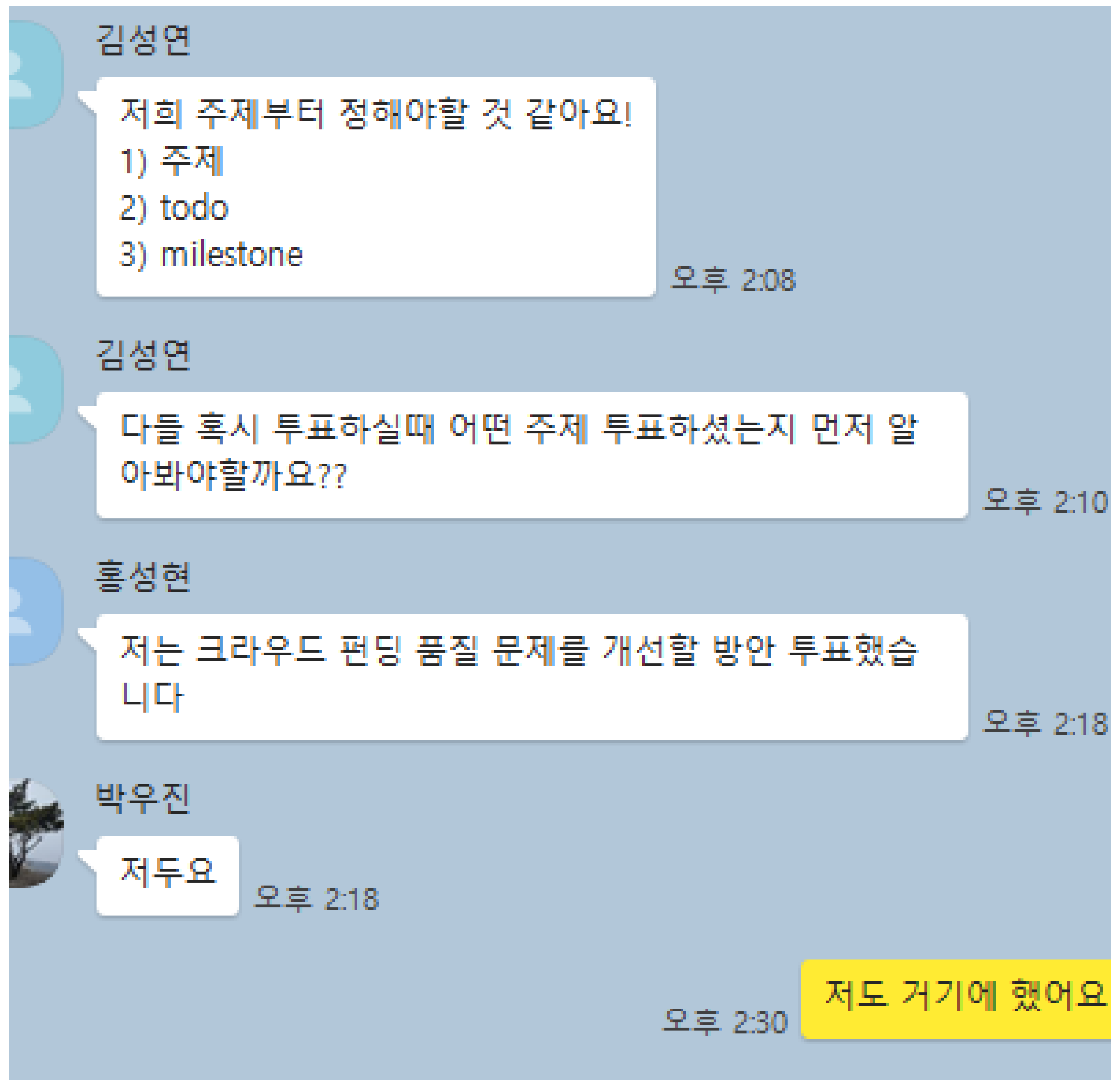
최근 19,000%라는 어마어마한 펀딩률을 달성한 이어폰이 있다. **놀랍게도 이 펀딩 프로젝트의 목표 금액은 50만원이다.** 최소 펀딩 가격은 약 8만원. 7명에게만 투자받아도 펀딩률

2. Malicious product

Many seller manipulate goal of funding amount, so user don't know exactly which ones are really popular one.

Motivation

Bigdata project - Crowd funding analysis



김성연

저희 주제부터 정해야할 것 같아요!
1) 주제
2) todo
3) milestone

오후 2:08

김성연

다들 혹시 투표하실때 어떤 주제 투표하셨는지 먼저 알아봐야할까요??

오후 2:10

홍성현

저는 크라우드 펀딩 품질 문제를 개선할 방안 투표했습니다

오후 2:18

박우진

저두요

오후 2:18

오후 2:30

저도 거기에 했어요

Uncomfortable!

So we think!

Motivation

Motivation

Bigdata project - Crowd funding analysis

Find user funding data

01

Get user funding data to analyze and recommend new product

02

Recommend

Using bigdata tech to recommend new product

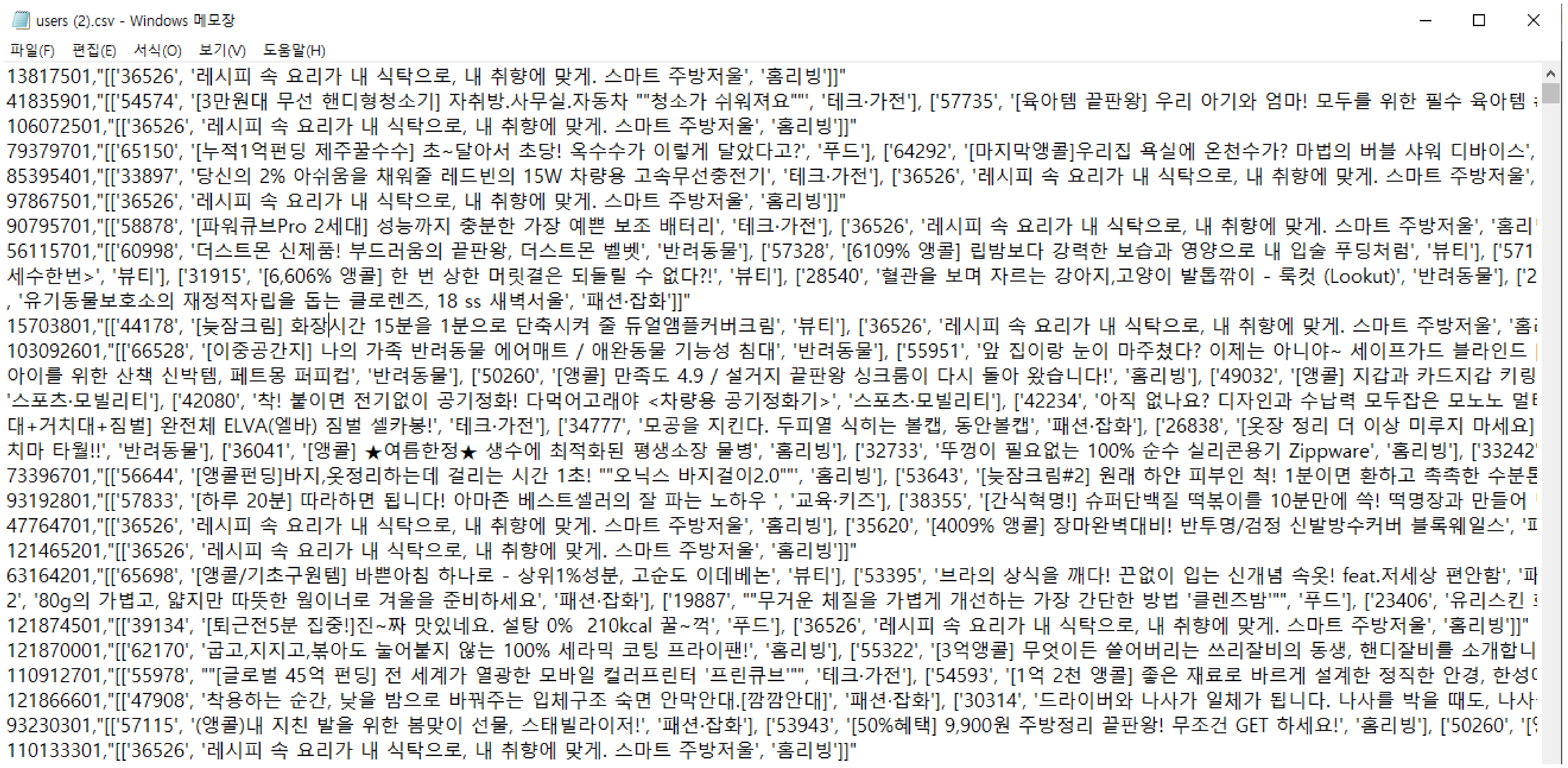
03

Result comparison

Using two recommend algorithm , and compare.

Requirements

Bigdata project - Crowd funding analysis

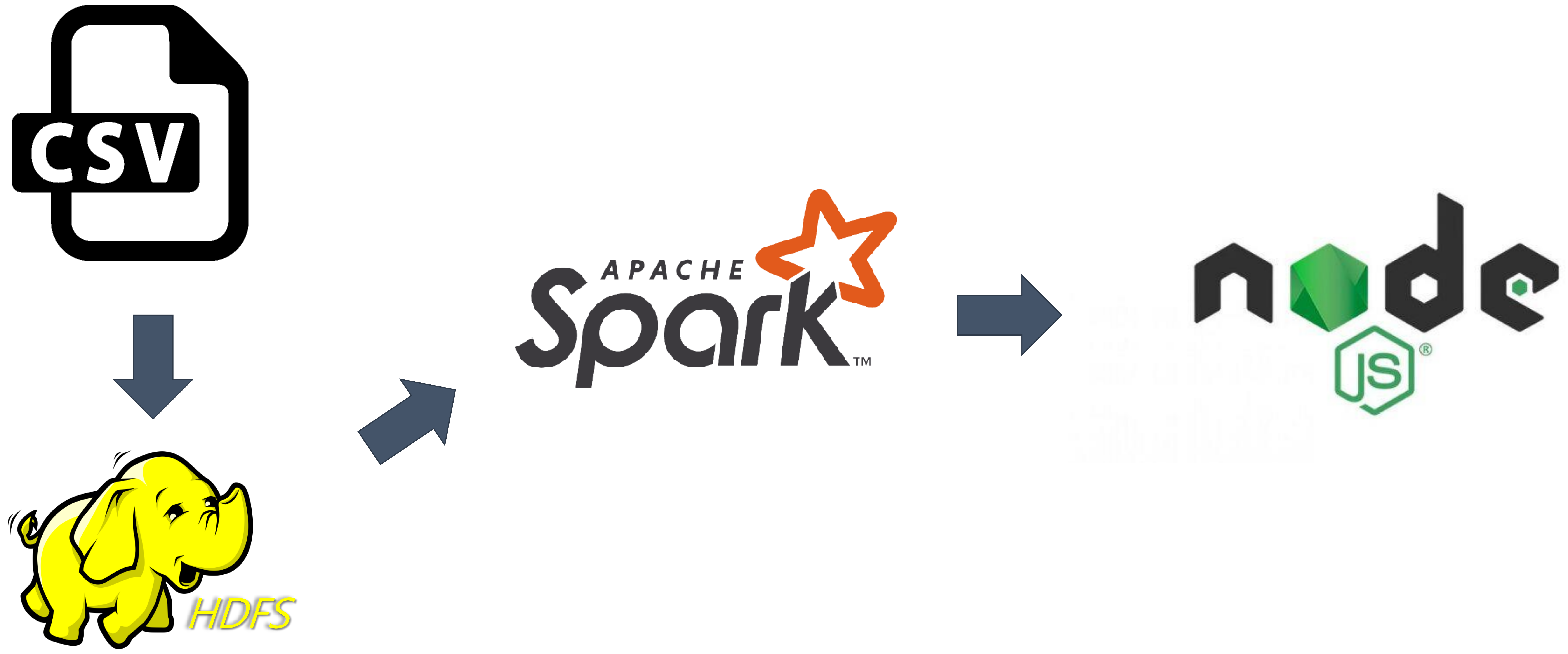

users (2).csv - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
13817501,"[['36526', '레시피 속 요리가 내 식탁으로, 내 취향에 맞게. 스마트 주방저울', '홈리빙']]"
41835901,"[['54574', '[3만원대 무선 핸디형청소기] 자취방.사무실.자동차 ""청소가 쉬워져요""', '테크·가전'], ['57735', '[육아템 끝판왕] 우리 아기와 엄마! 모두를 위한 필수 육아템
106072501,"[['36526', '레시피 속 요리가 내 식탁으로, 내 취향에 맞게. 스마트 주방저울', '홈리빙']]"
79379701,"[['65150', '[누적1억펀딩 제주꿀수수] 초~달아서 초당! 옥수수가 이렇게 달았다고?', '푸드'], ['64292', '[마지막앵콜]우리집 욕실에 온천수가? 마법의 버블 샤워 디바이스',
85395401,"[['33897', '당신의 2% 아쉬움을 채워줄 레드빈의 15W 차량용 고속무선충전기', '테크·가전'], ['36526', '레시피 속 요리가 내 식탁으로, 내 취향에 맞게. 스마트 주방저울',
97867501,"[['36526', '레시피 속 요리가 내 식탁으로, 내 취향에 맞게. 스마트 주방저울', '홈리빙']]"
90795701,"[['58878', '[파워큐브Pro 2세대] 성능까지 충분한 가장 예쁜 보조 배터리', '테크·가전'], ['36526', '레시피 속 요리가 내 식탁으로, 내 취향에 맞게. 스마트 주방저울', '홈리
56115701,"[['60998', '더스트몬 신제품! 부드러움의 끝판왕, 더스트몬 벨벳', '반려동물'], ['57328', '[6109% 앵콜] 립밤보다 강력한 보습과 영양으로 내 입술 푸딩처럼', '뷰티'], ['571
세수한번>', '뷰티'], ['31915', '[6,606% 앵콜] 한 번 상한 머릿결은 되돌릴 수 없다?!', '뷰티'], ['28540', '혈관을 보며 자르는 강아지,고양이 발톱깎이 - 룩컷 (Lookut)', '반려동물'], ['2
, '유기동물보호소의 재정적자립을 돕는 클로렌즈, 18 ss 새벽서울', '패션·잡화']]"
15703801,"[['44178', '[늦잠크림] 화장시간 15분을 1분으로 단축시켜 줄 듀얼앰플커버크림', '뷰티'], ['36526', '레시피 속 요리가 내 식탁으로, 내 취향에 맞게. 스마트 주방저울', '홈
103092601,"[['66528', '[이중공간지] 나의 가족 반려동물 에어매트 / 애완동물 기능성 침대', '반려동물'], ['55951', '앞 집이랑 눈이 마주쳤다? 이제는 아니야~ 세이프가드 블라인드
아이를 위한 산책 신박템, 페트몽 퍼피컵', '반려동물'], ['50260', '[앵콜] 만족도 4.9 / 설거지 끝판왕 싱크룸이 다시 돌아 왔습니다!', '홈리빙'], ['49032', '[앵콜] 지갑과 카드지갑 키링
'스포츠·모빌리티'], ['42080', '착! 붙이면 전기없이 공기정화! 다먹어고래야 <차량용 공기정화기>', '스포츠·모빌리티'], ['42234', '아직 없나요? 디자인과 수납력 모두잡은 모노노 멀
대+거치대+짐벌] 완전체 ELVA(엘바) 짐벌 셀카봉!', '테크·가전'], ['34777', '모공을 지킨다. 두피열 식히는 볼캡, 동안볼캡', '패션·잡화'], ['26838', '[옷장 정리 더 이상 미루지 마세요]
치마 타월!!', '반려동물'], ['36041', '[앵콜] ★여름한정★ 생수에 최적화된 평생소장 물병', '홈리빙'], ['32733', '뚜껑이 필요없는 100% 순수 실리콘용기 Zippware', '홈리빙'], ['33242'
73396701,"[['56644', '[앵콜펀딩]바지,옷정리하는데 걸리는 시간 1초! ""오닉스 바지걸이2.0""', '홈리빙'], ['53643', '[늦잠크림#2] 원래 하얀 피부인 척! 1분이면 환하고 촉촉한 수분
93192801,"[['57833', '[하루 20분] 따라하면 됩니다! 아마존 베스트셀러의 잘 파는 노하우 ', '교육·키즈'], ['38355', '[간식혁명!] 슈퍼단백질 떡볶이를 10분만에 쓱! 떡명장과 만들어
47764701,"[['36526', '레시피 속 요리가 내 식탁으로, 내 취향에 맞게. 스마트 주방저울', '홈리빙'], ['35620', '[4009% 앵콜] 장마완벽대비! 반투명/검정 신발방수커버 블록웨일스', '
121465201,"[['36526', '레시피 속 요리가 내 식탁으로, 내 취향에 맞게. 스마트 주방저울', '홈리빙']]"
63164201,"[['65698', '[앵콜/기초구원템] 바쁜아침 하나로 - 상위1%성분, 고순도 이데베논', '뷰티'], ['53395', '브라의 상식을 깨다! 끈없이 입는 신개념 속옷! feat.저세상 편안함', '파
2', '80g의 가볍고, 얇지만 따뜻한 웜이너로 겨울을 준비하세요', '패션·잡화'], ['19887', ""무거운 체질을 가볍게 개선하는 가장 간단한 방법 '클렌즈밤'""', '푸드'], ['23406', '유리스킨
121874501,"[['39134', '[퇴근전5분 집중!]진~짜 맛있네요. 설탕 0%  210kcal 꿀~꺽', '푸드'], ['36526', '레시피 속 요리가 내 식탁으로, 내 취향에 맞게. 스마트 주방저울', '홈리빙']]"
121870001,"[['62170', '굽고,지지고,볶아도 눌어붙지 않는 100% 세라믹 코팅 프라이팬!', '홈리빙'], ['55322', '[3억앵콜] 무엇이든 쓸어버리는 쓰리잘비의 동생, 핸디잘비를 소개합니
110912701,"[['55978', ""[글로벌 45억 펀딩] 전 세계가 열광한 모바일 컬러프린터 '프린큐브'""', '테크·가전'], ['54593', '[1억 2천 앵콜] 좋은 재료로 바르게 설계한 정직한 안경, 한성
121866601,"[['47908', '착용하는 순간, 낮을 밤으로 바꿔주는 입체구조 숙면 안막안대.[깜깜안대]', '패션·잡화'], ['30314', '드라이버와 나사가 일체가 됩니다. 나사를 박을 때도, 나사
93230301,"[['57115', '(앵콜)내 지친 발을 위한 봄맞이 선물, 스태빌라이저!', '패션·잡화'], ['53943', '[50%혜택] 9,900원 주방정리 끝판왕! 무조건 GET 하세요!', '홈리빙'], ['50260', '[
110133301,"[['36526', '레시피 속 요리가 내 식탁으로, 내 취향에 맞게. 스마트 주방저울', '홈리빙']]"
```

User funding data

Requirements

Bigdata project - Crowd funding analysis



Background

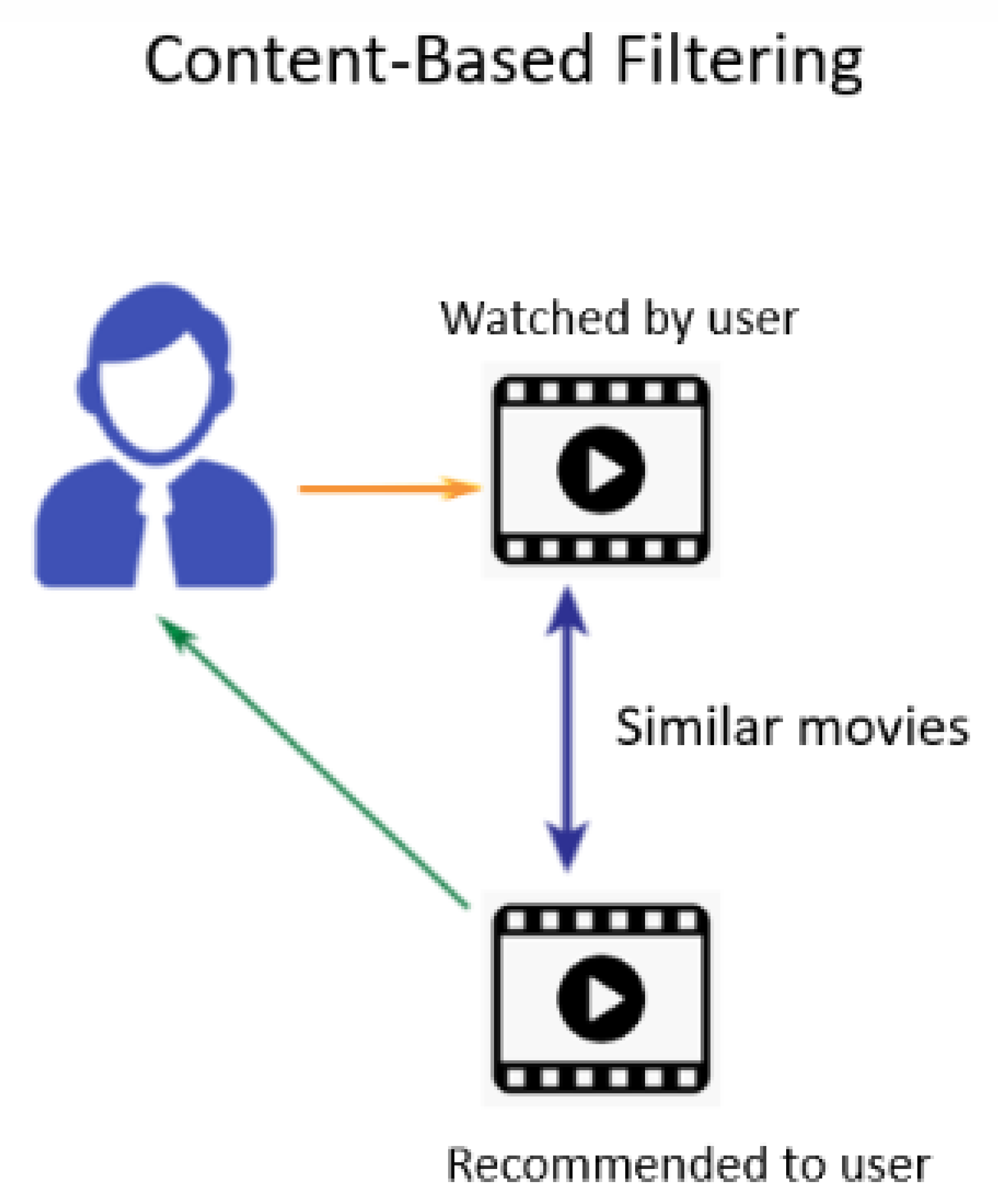
Bigdata project - Crowd funding analysis

For analysis

Background

Bigdata project - Crowd funding analysis

For visualization



Content based filtering

Content based filtering(CBF) is a method of recommendation system. When a user prefers a specific item, CBF will recommend another item that similar content to user's previous select to user.

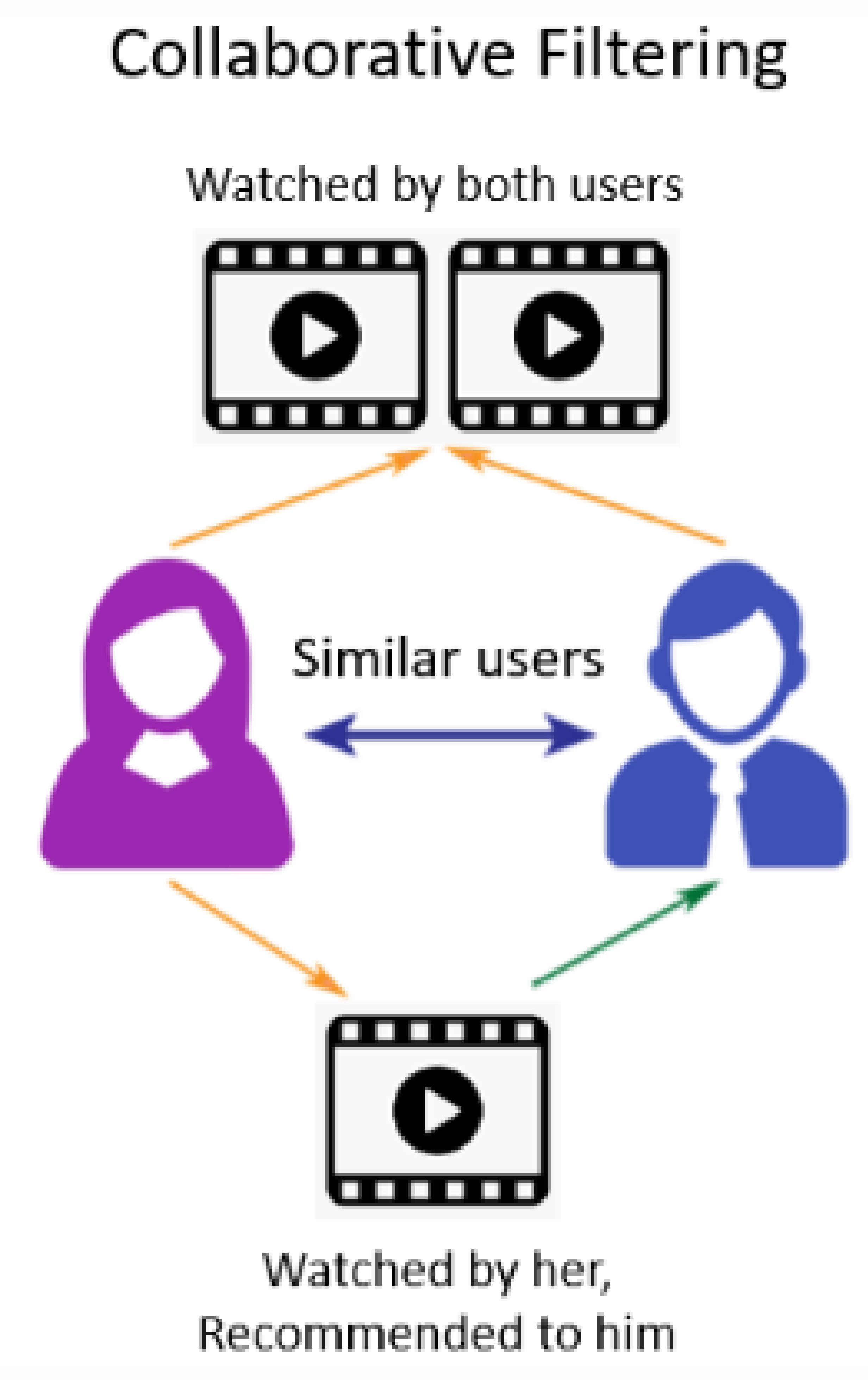
For example, if a user has watched Marvel's movie, based on this, another Marvel movie can be recommended by CBF algorithm.

Collaborative filtering

Collaborative filtering in the recommendation system refers to a technique that predicts itself based on taste information from many users.

Collaborative filtering is based on the fact that if a particular person a has the same opinion as person B on one issue, there is a high probability of having a similar opinion on other issues.

It similar to collective intelligence.



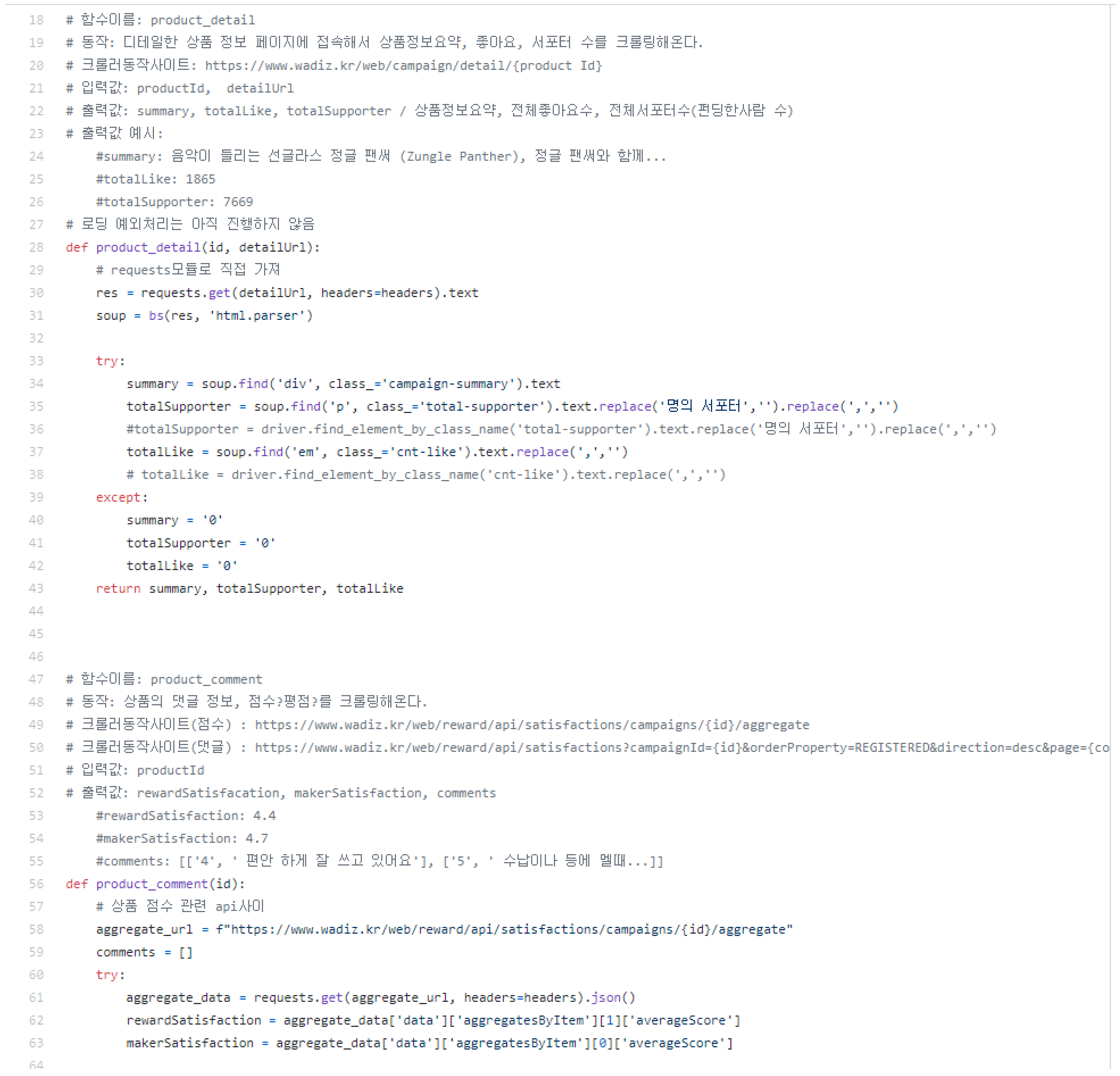
Data acquisition

Bigdata project - Crowd funding analysis

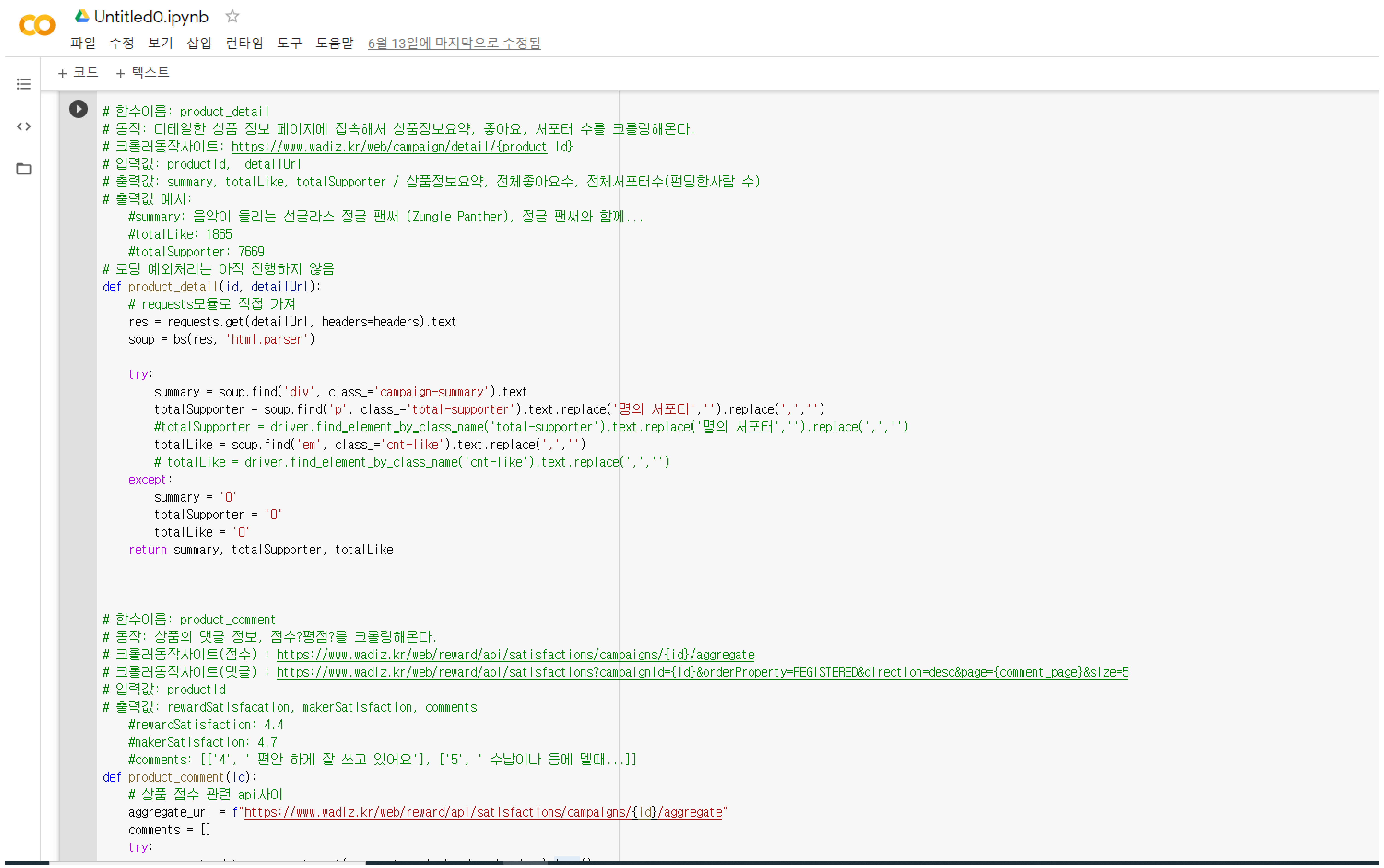
We want to use user comment data, but...

Data acquisition

Bigdata project - Crowd funding analysis



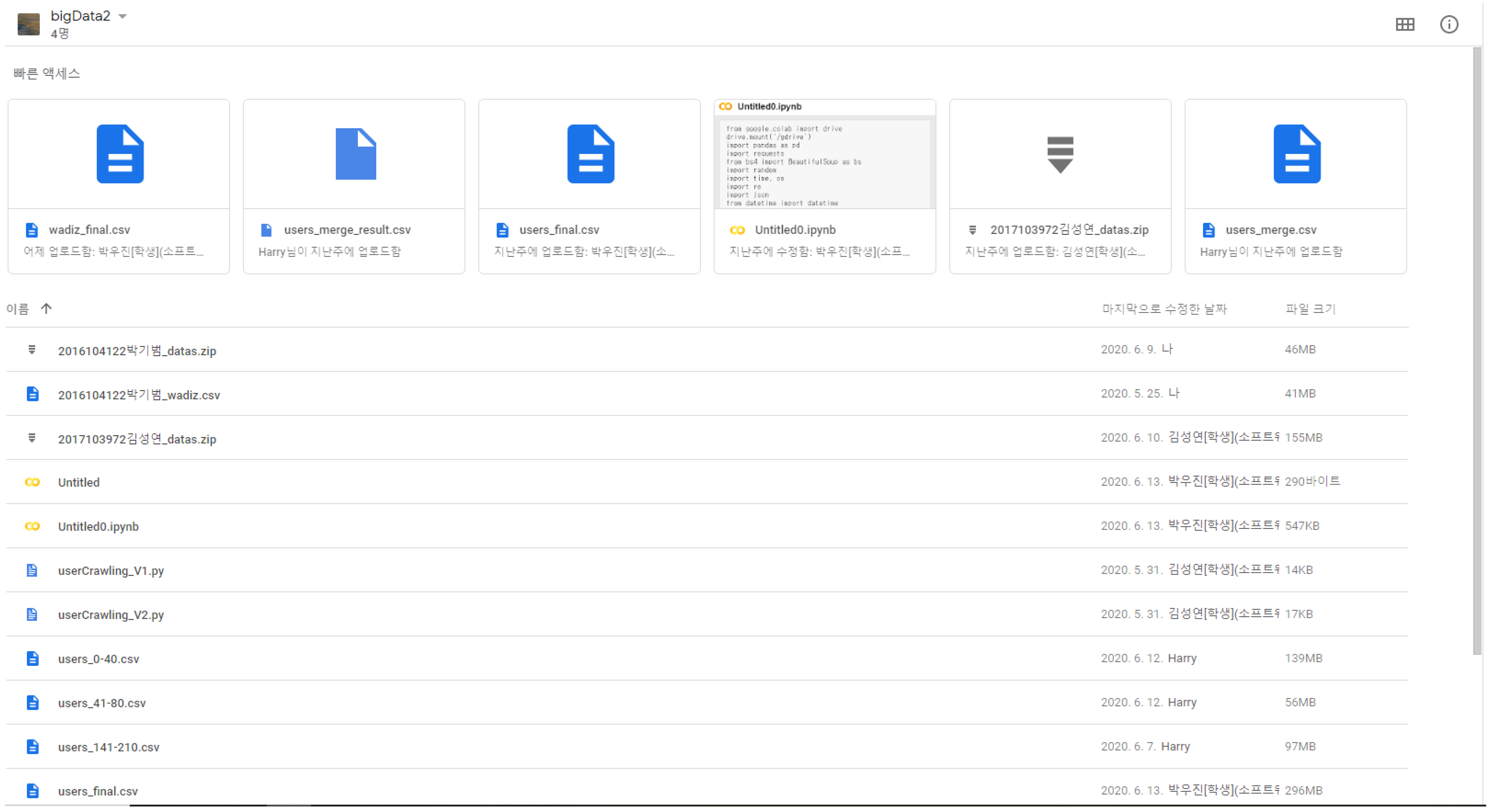
```python
# 함수이름: product_detail
# 동작: 디테일한 상품 정보 페이지에 접속해서 상품정보요약, 좋아요, 서포터 수를 크롤링해온다.
# 크롤러동작사이트: https://www.wadiz.kr/web/campaign/detail/{product Id}
# 입력값: productId,  detailUrl
# 출력값: summary, totalLike, totalSupporter / 상품정보요약, 전체좋아요수, 전체서포터수(펀딩한사람 수)
# 출력값 예시:
    #summary: 음악이 들리는 선글라스 정글 팬써 (Zungle Panther), 정글 팬써와 함께...
    #totalLike: 1865
    #totalSupporter: 7669
# 로딩 예외처리는 아직 진행하지 않음
def product_detail(id, detailUrl):
    # requests모듈로 직접 가져
    res = requests.get(detailUrl, headers=headers).text
    soup = bs(res, 'html.parser')

    try:
        summary = soup.find('div', class_='campaign-summary').text
        totalSupporter = soup.find('p', class_='total-supporter').text.replace('명의 서포터','').replace(',','')
        #totalSupporter = driver.find_element_by_class_name('total-supporter').text.replace('명의 서포터','').replace(',','')
        totalLike = soup.find('em', class_='cnt-like').text.replace(',','')
        # totalLike = driver.find_element_by_class_name('cnt-like').text.replace(',','')
    except:
        summary = '0'
        totalSupporter = '0'
        totalLike = '0'
    return summary, totalSupporter, totalLike



# 함수이름: product_comment
# 동작: 상품의 댓글 정보, 점수?평점?를 크롤링해온다.
# 크롤러동작사이트(점수) : https://www.wadiz.kr/web/reward/api/satisfactions/campaigns/{id}/aggregate
# 크롤러동작사이트(댓글) : https://www.wadiz.kr/web/reward/api/satisfactions?campaignId={id}&orderProperty=REGISTERED&direction=desc&page={co
# 입력값: productId
# 출력값: rewardSatisfacation, makerSatisfaction, comments
    #rewardSatisfaction: 4.4
    #makerSatisfaction: 4.7
    #comments: [['4', ' 편안 하게 잘 쓰고 있어요'], ['5', ' 수납이나 등에 멜때...]]
def product_comment(id):
    # 상품 점수 관련 api사이
    aggregate_url = f"https://www.wadiz.kr/web/reward/api/satisfactions/campaigns/{id}/aggregate"
    comments = []
    try:
        aggregate_data = requests.get(aggregate_url, headers=headers).json()
        rewardSatisfaction = aggregate_data['data']['aggregatesByItem'][1]['averageScore']
        makerSatisfaction = aggregate_data['data']['aggregatesByItem'][0]['averageScore']

```


CO
Untitled0.ipynb
파일 수정 보기 삽입 런타임 도구 도움말 6월 13일에 마지막으로 수정됨
+ 코드 + 텍스트

```python
# 함수이름: product_detail
# 동작: 디테일한 상품 정보 페이지에 접속해서 상품정보요약, 좋아요, 서포터 수를 크롤링해온다.
# 크롤러동작사이트: https://www.wadiz.kr/web/campaign/detail/{product Id}
# 입력값: productId,  detailUrl
# 출력값: summary, totalLike, totalSupporter / 상품정보요약, 전체좋아요수, 전체서포터수(펀딩한사람 수)
# 출력값 예시:
    #summary: 음악이 들리는 선글라스 정글 팬써 (Zungle Panther), 정글 팬써와 함께...
    #totalLike: 1865
    #totalSupporter: 7669
# 로딩 예외처리는 아직 진행하지 않음
def product_detail(id, detailUrl):
    # requests모듈로 직접 가져
    res = requests.get(detailUrl, headers=headers).text
    soup = bs(res, 'html.parser')

    try:
        summary = soup.find('div', class_='campaign-summary').text
        totalSupporter = soup.find('p', class_='total-supporter').text.replace('명의 서포터','').replace(',','')
        #totalSupporter = driver.find_element_by_class_name('total-supporter').text.replace('명의 서포터','').replace(',','')
        totalLike = soup.find('em', class_='cnt-like').text.replace(',','')
        # totalLike = driver.find_element_by_class_name('cnt-like').text.replace(',','')
    except:
        summary = '0'
        totalSupporter = '0'
        totalLike = '0'
    return summary, totalSupporter, totalLike



# 함수이름: product_comment
# 동작: 상품의 댓글 정보, 점수?평점?를 크롤링해온다.
# 크롤러동작사이트(점수) : https://www.wadiz.kr/web/reward/api/satisfactions/campaigns/{id}/aggregate
# 크롤러동작사이트(댓글) : https://www.wadiz.kr/web/reward/api/satisfactions?campaignId={id}&orderProperty=REGISTERED&direction=desc&page={comment_page}&size=5
# 입력값: productId
# 출력값: rewardSatisfacation, makerSatisfaction, comments
    #rewardSatisfaction: 4.4
    #makerSatisfaction: 4.7
    #comments: [['4', ' 편안 하게 잘 쓰고 있어요'], ['5', ' 수납이나 등에 멜때...]]
def product_comment(id):
    # 상품 점수 관련 api사이
    aggregate_url = f"https://www.wadiz.kr/web/reward/api/satisfactions/campaigns/{id}/aggregate"
    comments = []
    try:
```

Data acquisition

Bigdata project - Crowd funding analysis



bigData2
4명

빠른 액세스

wadiz_final.csv	어제 업로드함: 박우진[학생](소프트...
users_merge_result.csv	Harry님이 지난주에 업로드함
users_final.csv	지난주에 업로드함: 박우진[학생](소...
Untitled0.ipynb	지난주에 수정함: 박우진[학생](소프...
2017103972김성연_datas.zip	지난주에 업로드함: 김성연[학생](소...
users_merge.csv	Harry님이 지난주에 업로드함

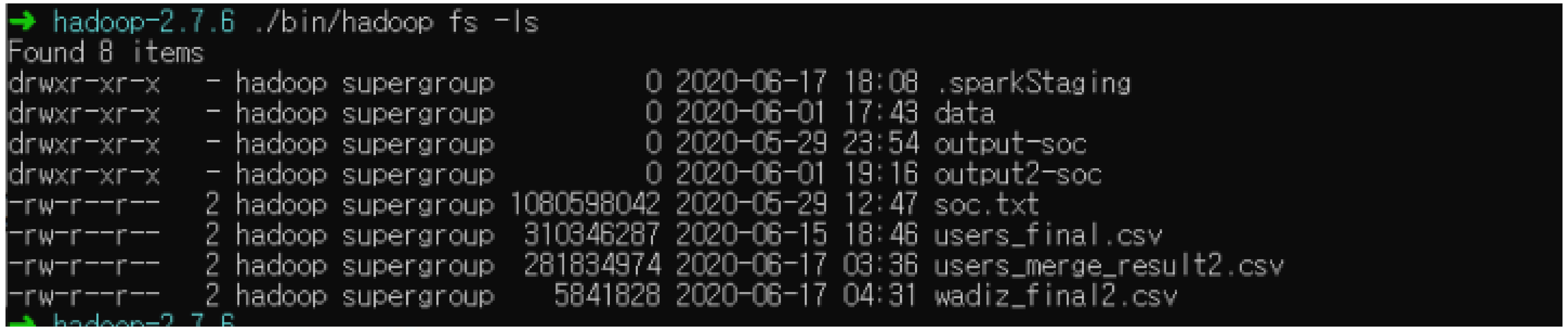
Untitled0.ipynb

```
from google.colab import drive
drive.mount('/gdrive')
import pandas as pd
import requests
from bs4 import BeautifulSoup as bs
import random
import time, os
import re
import json
from datetime import datetime
```

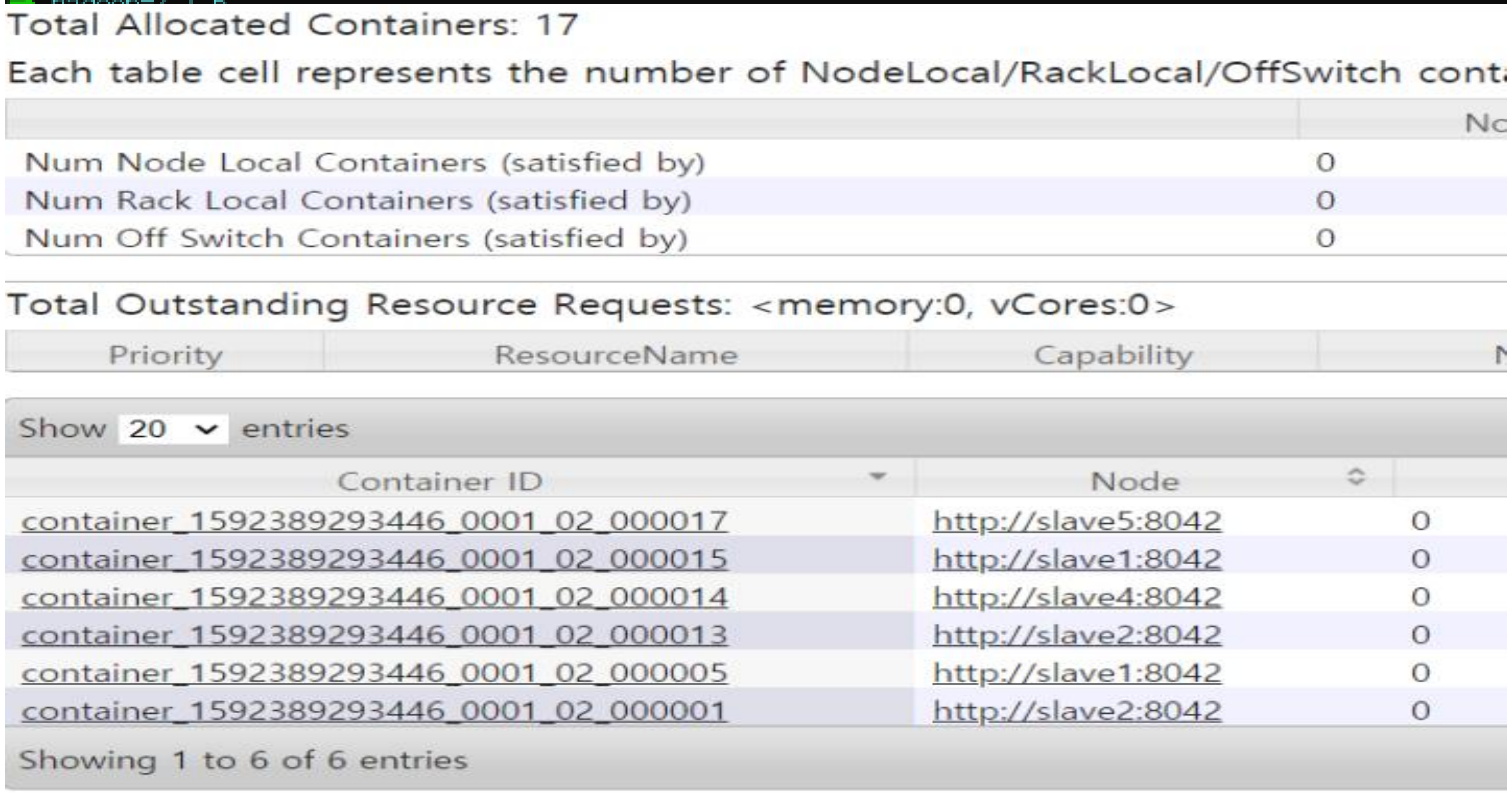
이름 ↑	마지막으로 수정한 날짜	파일 크기
2016104122박기범_datas.zip	2020. 6. 9. 나	46MB
2016104122박기범_wadiz.csv	2020. 5. 25. 나	41MB
2017103972김성연_datas.zip	2020. 6. 10. 김성연[학생](소프트	155MB
Untitled	2020. 6. 13. 박우진[학생](소프트	290바이트
Untitled0.ipynb	2020. 6. 13. 박우진[학생](소프트	547KB
userCrawling_V1.py	2020. 5. 31. 김성연[학생](소프트	14KB
userCrawling_V2.py	2020. 5. 31. 김성연[학생](소프트	17KB
users_0-40.csv	2020. 6. 12. Harry	139MB
users_41-80.csv	2020. 6. 12. Harry	56MB
users_141-210.csv	2020. 6. 7. Harry	97MB
users_final.csv	2020. 6. 13. 박우진[학생](소프트	296MB

Data Storage

Bigdata project - Crowd funding analysis



```
➜ hadoop-2.7.6 ./bin/hadoop fs -ls
Found 8 items
drwxr-xr-x   - hadoop supergroup          0 2020-06-17 18:08 .sparkStaging
drwxr-xr-x   - hadoop supergroup          0 2020-06-01 17:43 data
drwxr-xr-x   - hadoop supergroup          0 2020-05-29 23:54 output-soc
drwxr-xr-x   - hadoop supergroup          0 2020-06-01 19:16 output2-soc
-rw-r--r--   2 hadoop supergroup 1080598042 2020-05-29 12:47 soc.txt
-rw-r--r--   2 hadoop supergroup  310346287 2020-06-15 18:46 users_final.csv
-rw-r--r--   2 hadoop supergroup  281834974 2020-06-17 03:36 users_merge_result2.csv
-rw-r--r--   2 hadoop supergroup    5841828 2020-06-17 04:31 wadiz_final2.csv
➜ hadoop-2.7.6
```



Total Allocated Containers: 17

Each table cell represents the number of NodeLocal/RackLocal/OffSwitch cont

	No
Num Node Local Containers (satisfied by)	0
Num Rack Local Containers (satisfied by)	0
Num Off Switch Containers (satisfied by)	0

Total Outstanding Resource Requests: <memory:0, vCores:0>

Priority	ResourceName	Capability	N

Show 20 entries

Container ID	Node	
container_1592389293446_0001_02_000017	http://slave5:8042	0
container_1592389293446_0001_02_000015	http://slave1:8042	0
container_1592389293446_0001_02_000014	http://slave4:8042	0
container_1592389293446_0001_02_000013	http://slave2:8042	0
container_1592389293446_0001_02_000005	http://slave1:8042	0
container_1592389293446_0001_02_000001	http://slave2:8042	0

Showing 1 to 6 of 6 entries

Data analysis

Bigdata project - Crowd funding analysis

01 **Analysis using content-based filtering**

Analysis user funding data with CBF algorithm and display the result on web page

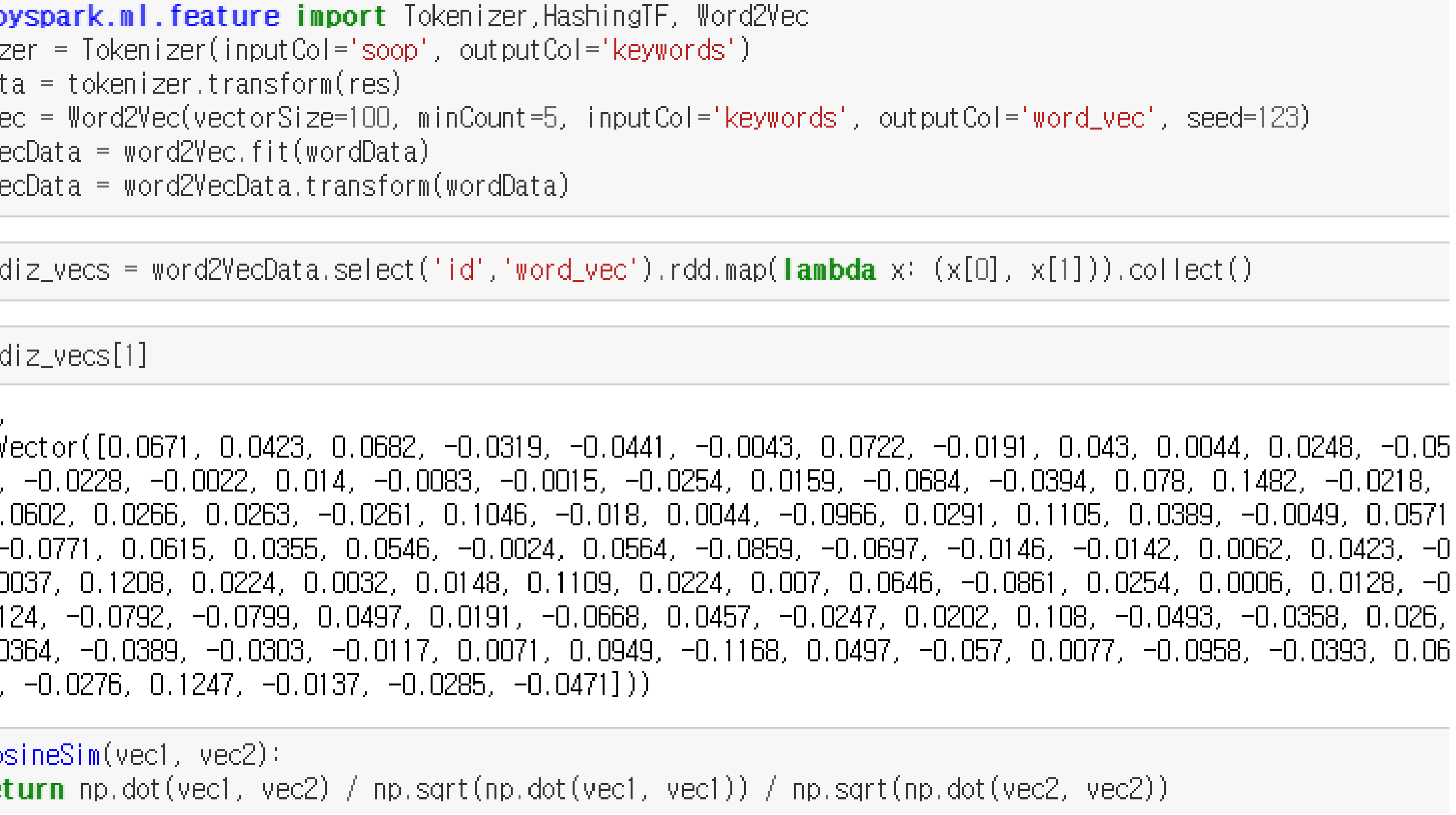
02 **Analysis using collaborative filtering**

Analysis user funding data with CF algorithm and display the result on web page

03 **Compare two result**

We can compare two result made by CBF and CF algorithm.

Using CBF algorithm



```
yspark.ml.feature import Tokenizer,HashingTF, Word2Vec
zer = Tokenizer(inputCol='soop', outputCol='keywords')
ta = tokenizer.transform(res)
ec = Word2Vec(vectorSize=100, minCount=5, inputCol='keywords', outputCol='word_vec', seed=123)
ecData = word2Vec.fit(wordData)
ecData = word2VecData.transform(wordData)
```

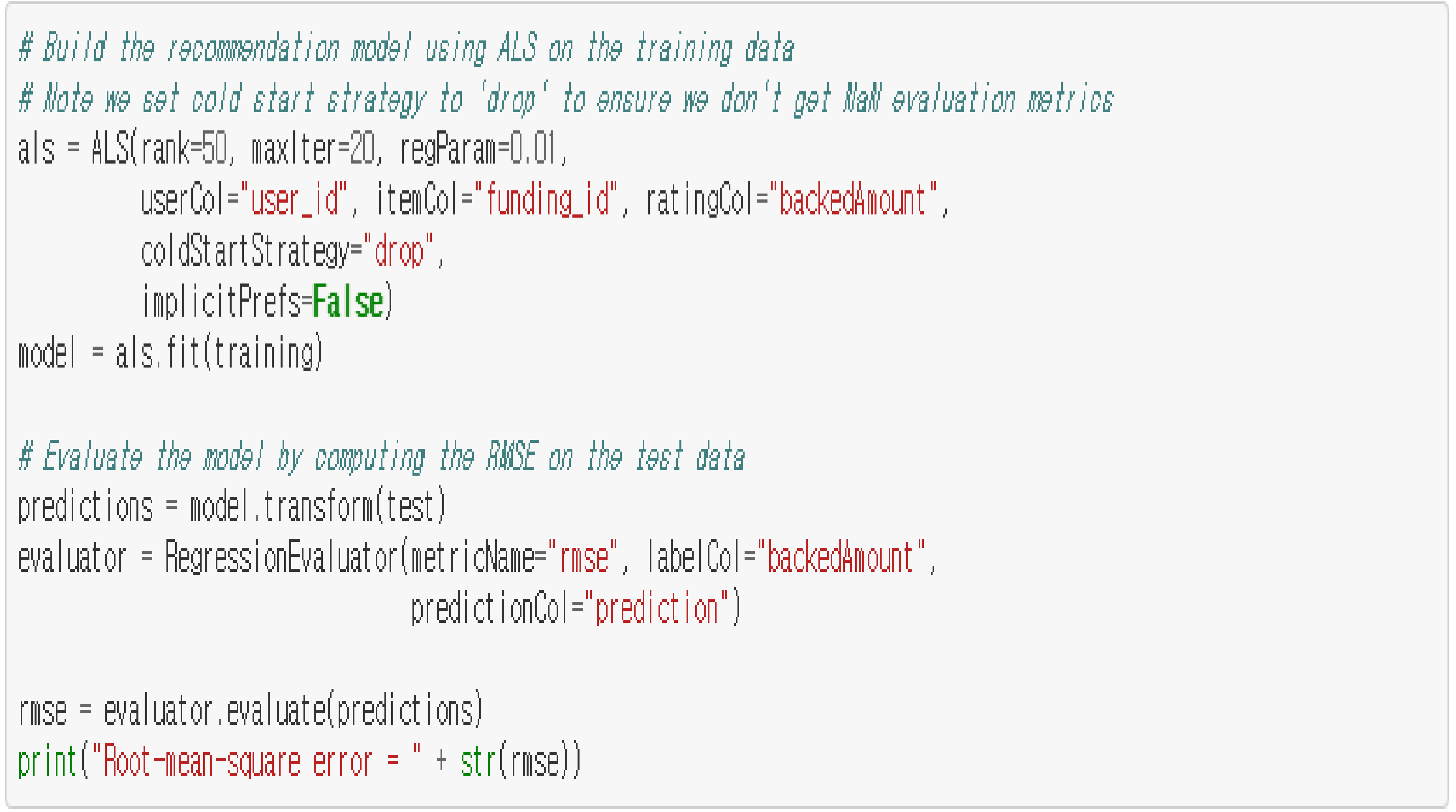
```
diz_vecs = word2VecData.select('id','word_vec').rdd.map(lambda x: (x[0], x[1])).collect()
```

```
diz_vecs[1]
```

```
,
Vector([0.0671, 0.0423, 0.0682, -0.0319, -0.0441, -0.0043, 0.0722, -0.0191, 0.043, 0.0044, 0.0248, -0.05
, -0.0228, -0.0022, 0.014, -0.0083, -0.0015, -0.0254, 0.0159, -0.0684, -0.0394, 0.078, 0.1482, -0.0218,
.0602, 0.0266, 0.0263, -0.0261, 0.1046, -0.018, 0.0044, -0.0966, 0.0291, 0.1105, 0.0389, -0.0049, 0.0571
-0.0771, 0.0615, 0.0355, 0.0546, -0.0024, 0.0564, -0.0859, -0.0697, -0.0146, -0.0142, 0.0062, 0.0423, -0
0037, 0.1208, 0.0224, 0.0032, 0.0148, 0.1109, 0.0224, 0.007, 0.0646, -0.0861, 0.0254, 0.0006, 0.0128, -0
124, -0.0792, -0.0799, 0.0497, 0.0191, -0.0668, 0.0457, -0.0247, 0.0202, 0.108, -0.0493, -0.0358, 0.026,
0364, -0.0389, -0.0303, -0.0117, 0.0071, 0.0949, -0.1168, 0.0497, -0.057, 0.0077, -0.0958, -0.0393, 0.06
, -0.0276, 0.1247, -0.0137, -0.0285, -0.0471]))
```

```
osineSim(vec1, vec2):
turn np.dot(vec1, vec2) / np.sqrt(np.dot(vec1, vec1)) / np.sqrt(np.dot(vec2, vec2))
```

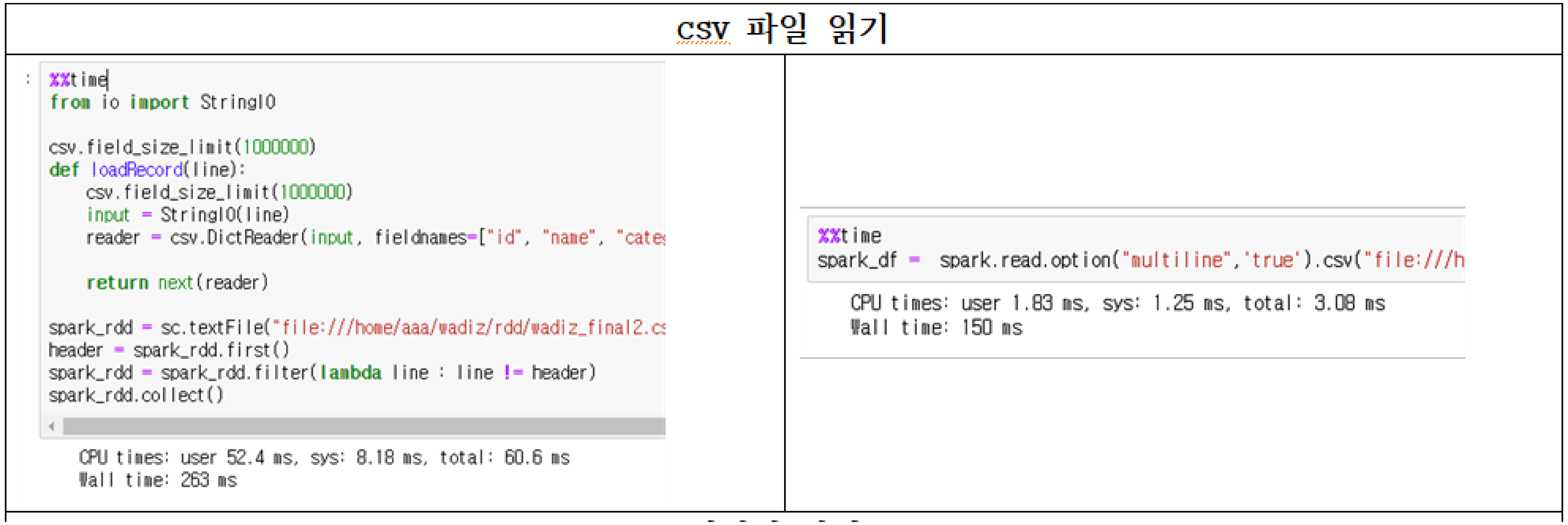
Using CF algorithm



```
# Build the recommendation model using ALS on the training data
# Note we set cold start strategy to 'drop' to ensure we don't get NaN evaluation metrics
als = ALS(rank=50, maxIter=20, regParam=0.01,
          userCol="user_id", itemCol="funding_id", ratingCol="backedAmount",
          coldStartStrategy="drop",
          implicitPrefs=False)
model = als.fit(training)

# Evaluate the model by computing the RMSE on the test data
predictions = model.transform(test)
evaluator = RegressionEvaluator(metricName="rmse", labelCol="backedAmount",
                                predictionCol="prediction")

rmse = evaluator.evaluate(predictions)
print("Root-mean-square error = " + str(rmse))
```

Data analysis - CBF

Bigdata project - Crowd funding analysis



csv 파일 읽기

```
%%time
from io import StringIO

csv.field_size_limit(1000000)
def loadRecord(line):
    csv.field_size_limit(1000000)
    input = StringIO(line)
    reader = csv.DictReader(input, fieldnames=["id", "name", "cate

    return next(reader)

spark_rdd = sc.textFile("file:///home/aaa/wadiz/rdd/wadiz_final2.cs
header = spark_rdd.first()
spark_rdd = spark_rdd.filter(lambda line : line != header)
spark_rdd.collect()
```

```
CPU times: user 52.4 ms, sys: 8.18 ms, total: 60.6 ms
Wall time: 263 ms
```

```
%%time
spark_df =  spark.read.option("multiline",'true').csv("file:///h
```

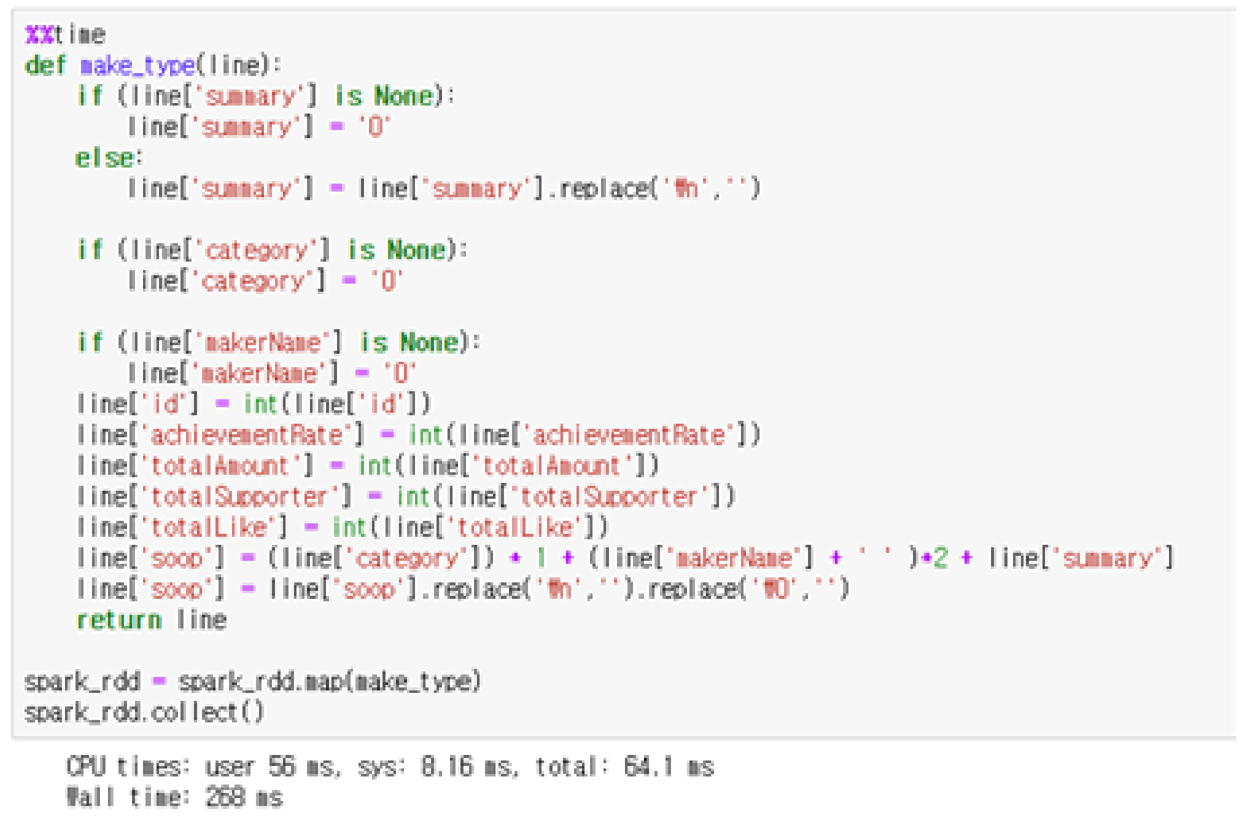
```
CPU times: user 1.83 ms, sys: 1.25 ms, total: 3.08 ms
Wall time: 150 ms
```

CSV read

Data analysis - CBF

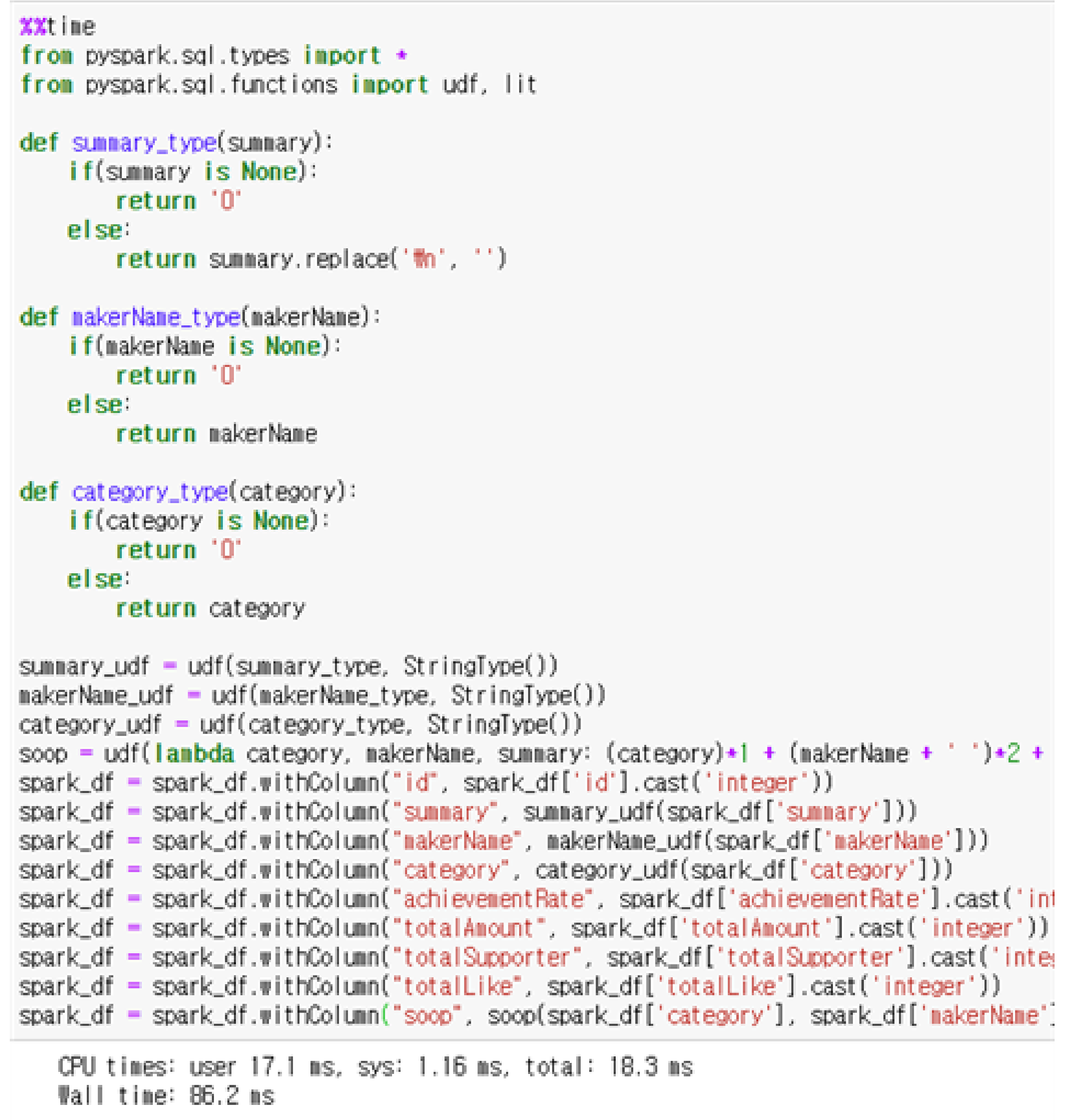
Bigdata project - Crowd funding analysis

데이터 정제1



```
%%time
def make_type(line):
    if (line['summary'] is None):
        line['summary'] = '0'
    else:
        line['summary'] = line['summary'].replace('₩n','')

    if (line['category'] is None):
        line['category'] = '0'

    if (line['makerName'] is None):
        line['makerName'] = '0'
    line['id'] = int(line['id'])
    line['achievementRate'] = int(line['achievementRate'])
    line['totalAmount'] = int(line['totalAmount'])
    line['totalSupporter'] = int(line['totalSupporter'])
    line['totalLike'] = int(line['totalLike'])
    line['soop'] = (line['category']) * 1 + (line['makerName'] + ' ' )*2 + line['summary']
    line['soop'] = line['soop'].replace('₩n','').replace('₩0','')
    return line

spark_rdd = spark_rdd.map(make_type)
spark_rdd.collect()
```

```
    CPU times: user 56 ms, sys: 8.16 ms, total: 64.1 ms
    Wall time: 268 ms
```



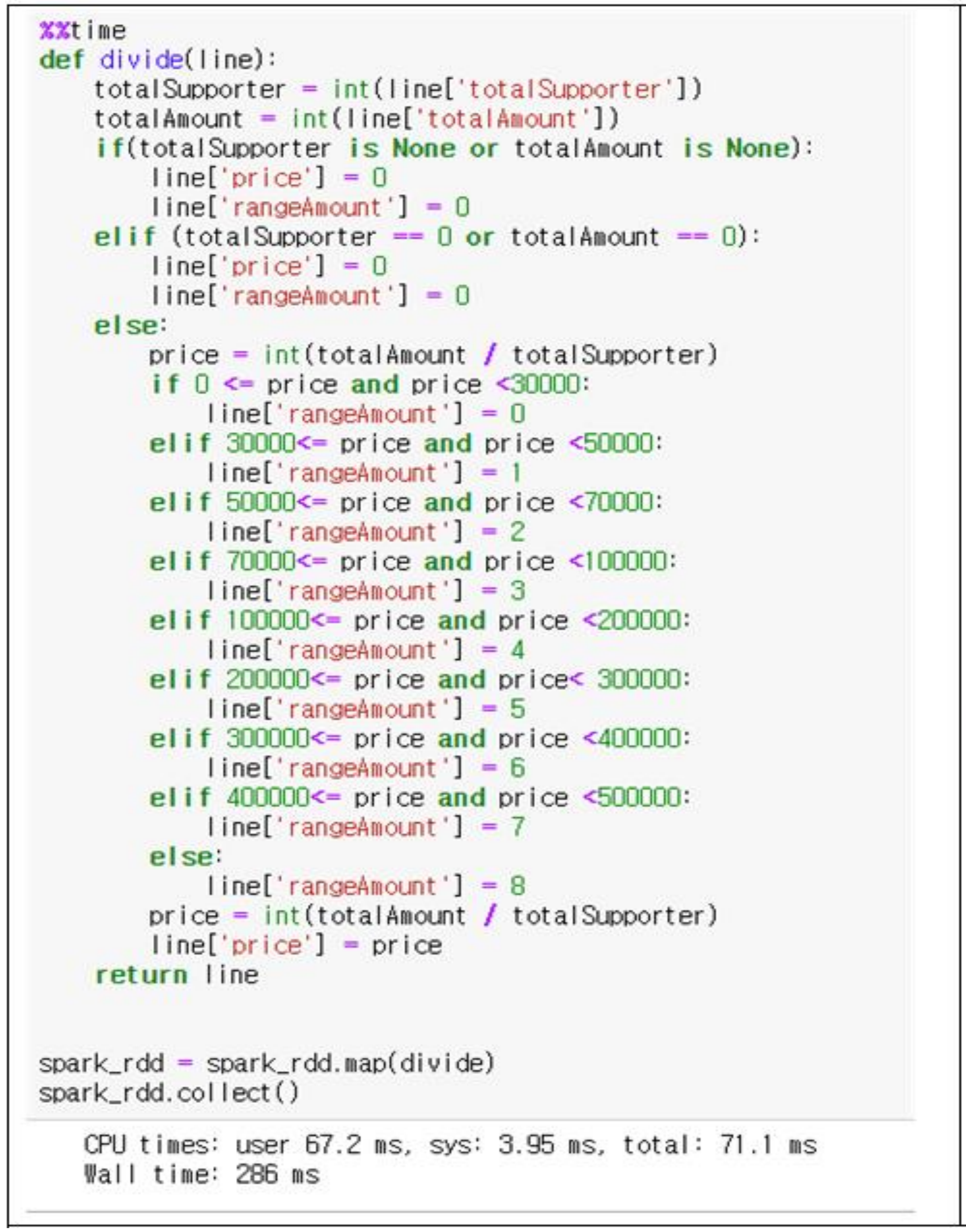
```
%%time
from pyspark.sql.types import *
from pyspark.sql.functions import udf, lit

def summary_type(summary):
    if(summary is None):
        return '0'
    else:
        return summary.replace('₩n', '')

def makerName_type(makerName):
    if(makerName is None):
        return '0'
    else:
        return makerName

def category_type(category):
    if(category is None):
        return '0'
    else:
        return category

summary_udf = udf(summary_type, StringType())
makerName_udf = udf(makerName_type, StringType())
category_udf = udf(category_type, StringType())
soop = udf(lambda category, makerName, summary: (category)*1 + (makerName + ' ')*2 +
spark_df = spark_df.withColumn("id", spark_df['id'].cast('integer'))
spark_df = spark_df.withColumn("summary", summary_udf(spark_df['summary']))
spark_df = spark_df.withColumn("makerName", makerName_udf(spark_df['makerName']))
spark_df = spark_df.withColumn("category", category_udf(spark_df['category']))
spark_df = spark_df.withColumn("achievementRate", spark_df['achievementRate'].cast('in
spark_df = spark_df.withColumn("totalAmount", spark_df['totalAmount'].cast('integer'))
spark_df = spark_df.withColumn("totalSupporter", spark_df['totalSupporter'].cast('inte
spark_df = spark_df.withColumn("totalLike", spark_df['totalLike'].cast('integer'))
spark_df = spark_df.withColumn("soop", soop(spark_df['category'], spark_df['makerName']
```

```
    CPU times: user 17.1 ms, sys: 1.16 ms, total: 18.3 ms
    Wall time: 86.2 ms
```

Make data frame and RDD using user data

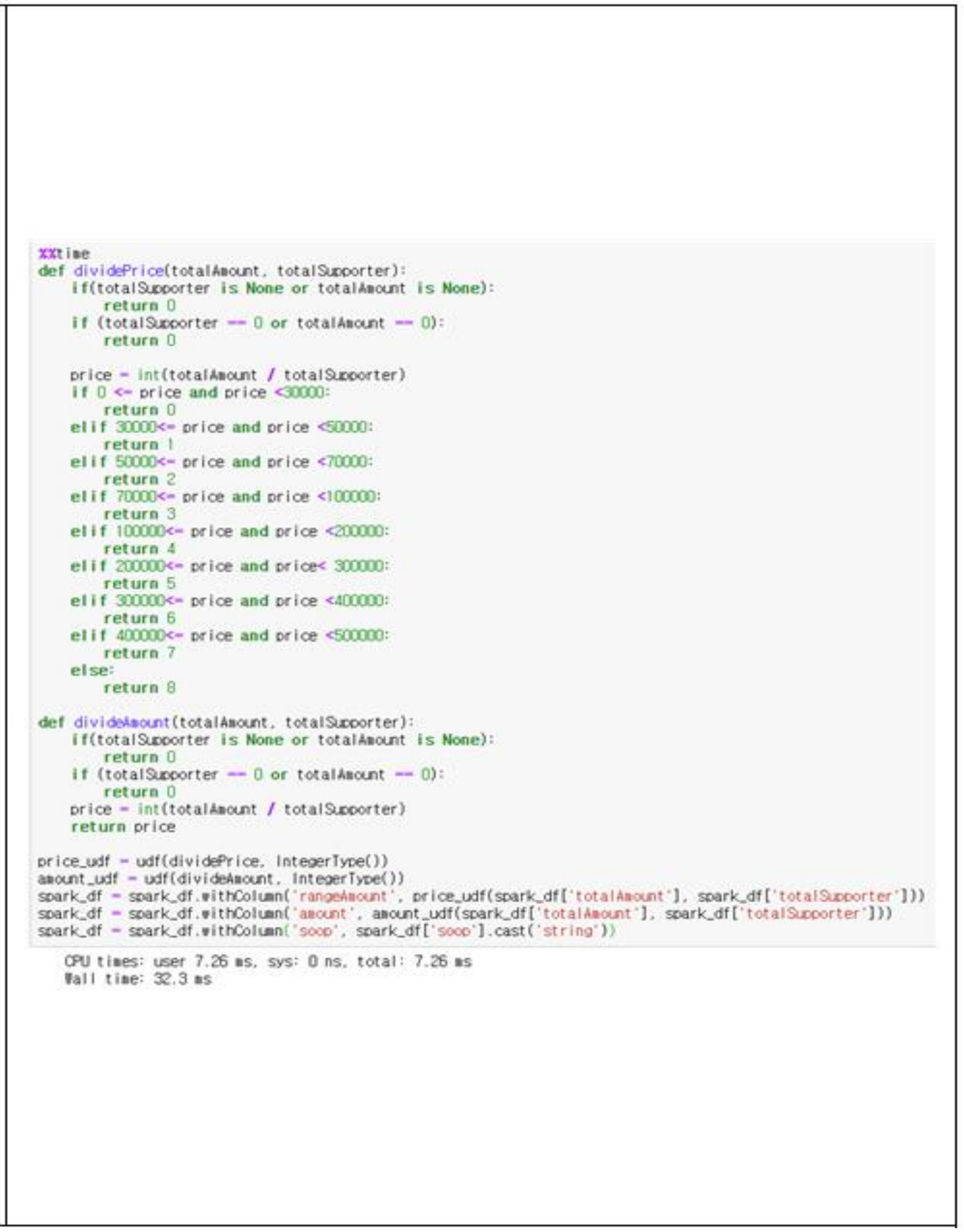
Data analysis - CBF

Bigdata project - Crowd funding analysis



```
%%time
def divide(line):
    totalSupporter = int(line['totalSupporter'])
    totalAmount = int(line['totalAmount'])
    if(totalSupporter is None or totalAmount is None):
        line['price'] = 0
        line['rangeAmount'] = 0
    elif (totalSupporter == 0 or totalAmount == 0):
        line['price'] = 0
        line['rangeAmount'] = 0
    else:
        price = int(totalAmount / totalSupporter)
        if 0 <= price and price <30000:
            line['rangeAmount'] = 0
        elif 30000<= price and price <50000:
            line['rangeAmount'] = 1
        elif 50000<= price and price <70000:
            line['rangeAmount'] = 2
        elif 70000<= price and price <100000:
            line['rangeAmount'] = 3
        elif 100000<= price and price <200000:
            line['rangeAmount'] = 4
        elif 200000<= price and price< 300000:
            line['rangeAmount'] = 5
        elif 300000<= price and price <400000:
            line['rangeAmount'] = 6
        elif 400000<= price and price <500000:
            line['rangeAmount'] = 7
        else:
            line['rangeAmount'] = 8
        price = int(totalAmount / totalSupporter)
        line['price'] = price
    return line


spark_rdd = spark_rdd.map(divide)
spark_rdd.collect()
```

```
CPU times: user 67.2 ms, sys: 3.95 ms, total: 71.1 ms
Wall time: 286 ms
```

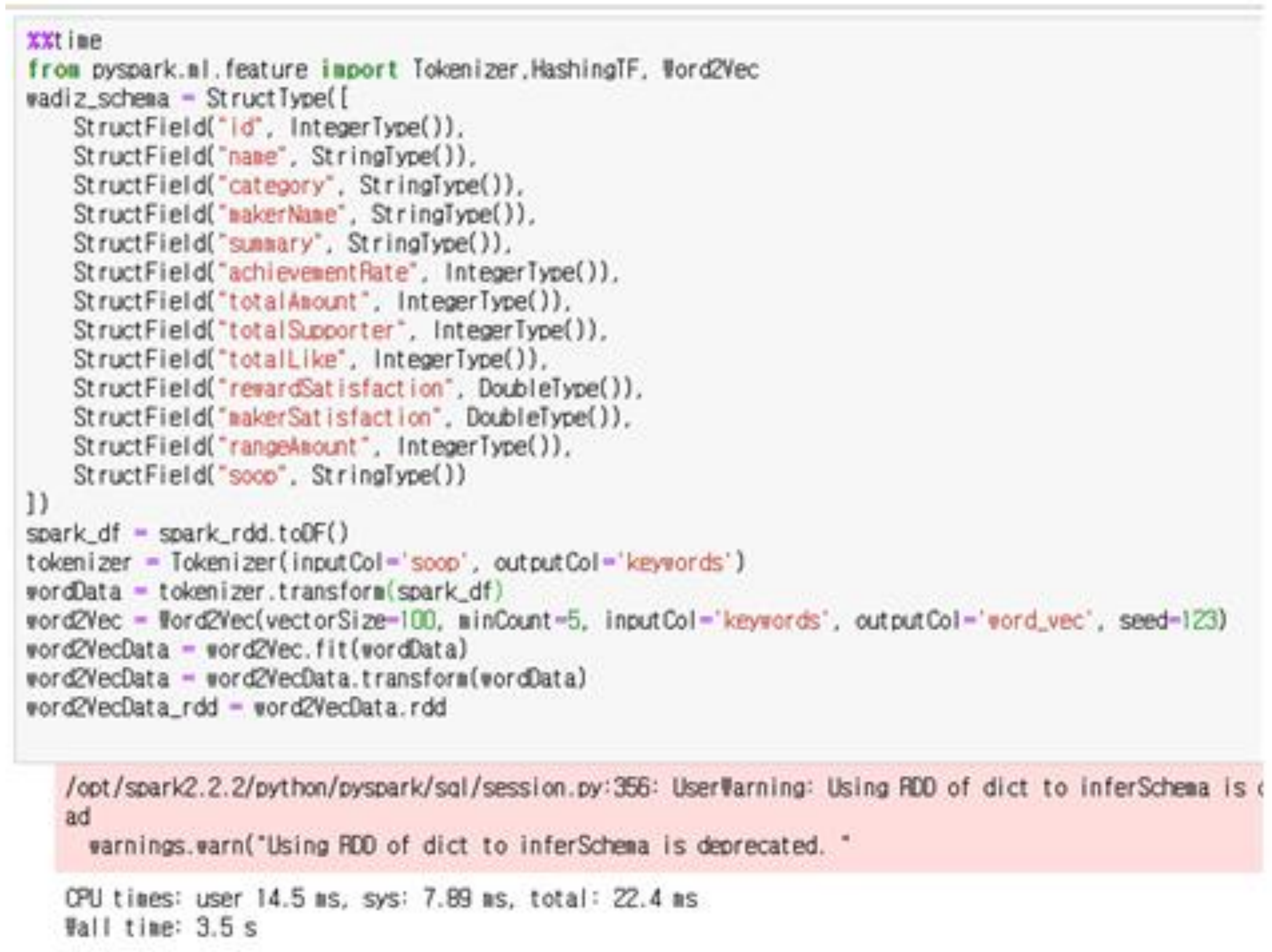


```
%%time
def dividePrice(totalAmount, totalSupporter):
    if(totalSupporter is None or totalAmount is None):
        return 0
    if (totalSupporter == 0 or totalAmount == 0):
        return 0

    price = int(totalAmount / totalSupporter)
    if 0 <= price and price <30000:
        return 0
    elif 30000<= price and price <50000:
        return 1
    elif 50000<= price and price <70000:
        return 2
    elif 70000<= price and price <100000:
        return 3
    elif 100000<= price and price <200000:
        return 4
    elif 200000<= price and price< 300000:
        return 5
    elif 300000<= price and price <400000:
        return 6
    elif 400000<= price and price <500000:
        return 7
    else:
        return 8

def divideAmount(totalAmount, totalSupporter):
    if(totalSupporter is None or totalAmount is None):
        return 0
    if (totalSupporter == 0 or totalAmount == 0):
        return 0
    price = int(totalAmount / totalSupporter)
    return price

price_udf = udf(dividePrice, IntegerType())
amount_udf = udf(divideAmount, IntegerType())
spark_df = spark_df.withColumn('rangeAmount', price_udf(spark_df['totalAmount'], spark_df['totalSupporter']))
spark_df = spark_df.withColumn('amount', amount_udf(spark_df['totalAmount'], spark_df['totalSupporter']))
spark_df = spark_df.withColumn('soop', spark_df['soop'].cast('string'))
```

```
CPU times: user 7.26 ms, sys: 0 ns, total: 7.26 ms
Wall time: 32.3 ms
```

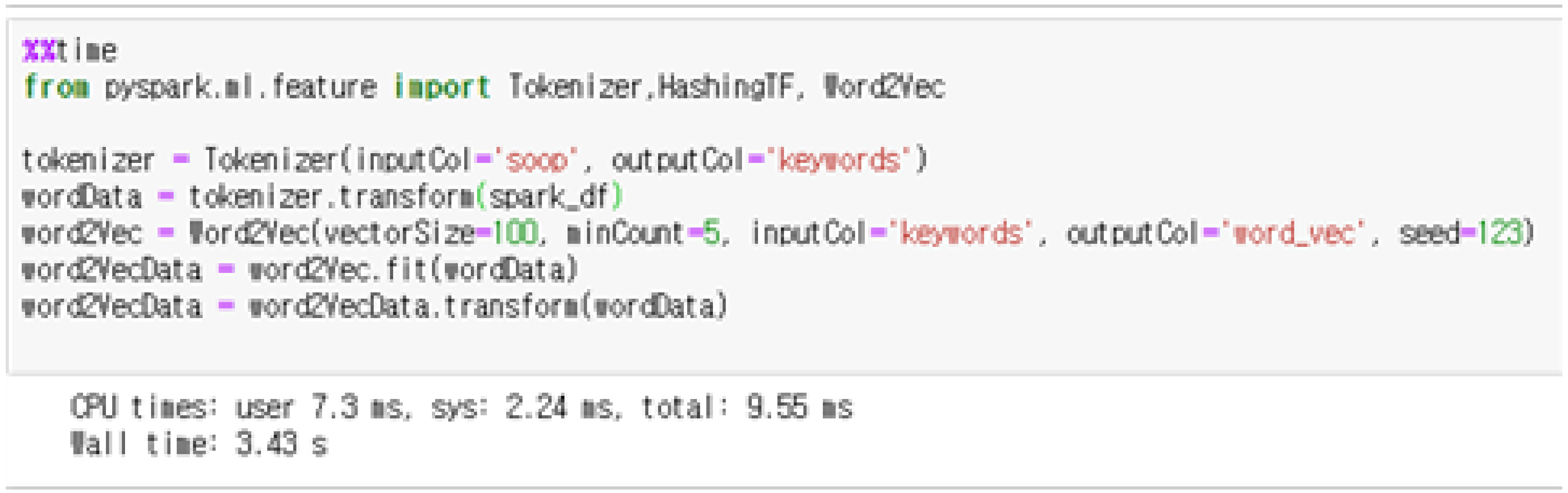
Data analysis - CBF

Bigdata project - Crowd funding analysis



```
%%time
from pyspark.ml.feature import Tokenizer,HashingTF, Word2Vec
wadiz_schema = StructType([
    StructField("id", IntegerType()),
    StructField("name", StringType()),
    StructField("category", StringType()),
    StructField("makerName", StringType()),
    StructField("summary", StringType()),
    StructField("achievementRate", IntegerType()),
    StructField("totalAmount", IntegerType()),
    StructField("totalSupporter", IntegerType()),
    StructField("totalLike", IntegerType()),
    StructField("rewardSatisfaction", DoubleType()),
    StructField("makerSatisfaction", DoubleType()),
    StructField("rangeAmount", IntegerType()),
    StructField("soop", StringType())
])
spark_df = spark_rdd.toDF()
tokenizer = Tokenizer(inputCol='soop', outputCol='keywords')
wordData = tokenizer.transform(spark_df)
word2Vec = Word2Vec(vectorSize=100, minCount=5, inputCol='keywords', outputCol='word_vec', seed=123)
word2VecData = word2Vec.fit(wordData)
word2VecData = word2VecData.transform(wordData)
word2VecData_rdd = word2VecData.rdd
```

```
/opt/spark2.2.2/python/pyspark/sql/session.py:356: UserWarning: Using RDD of dict to inferSchema is 
ad
  warnings.warn("Using RDD of dict to inferSchema is deprecated. "

CPU times: user 14.5 ms, sys: 7.89 ms, total: 22.4 ms
Wall time: 3.5 s
```



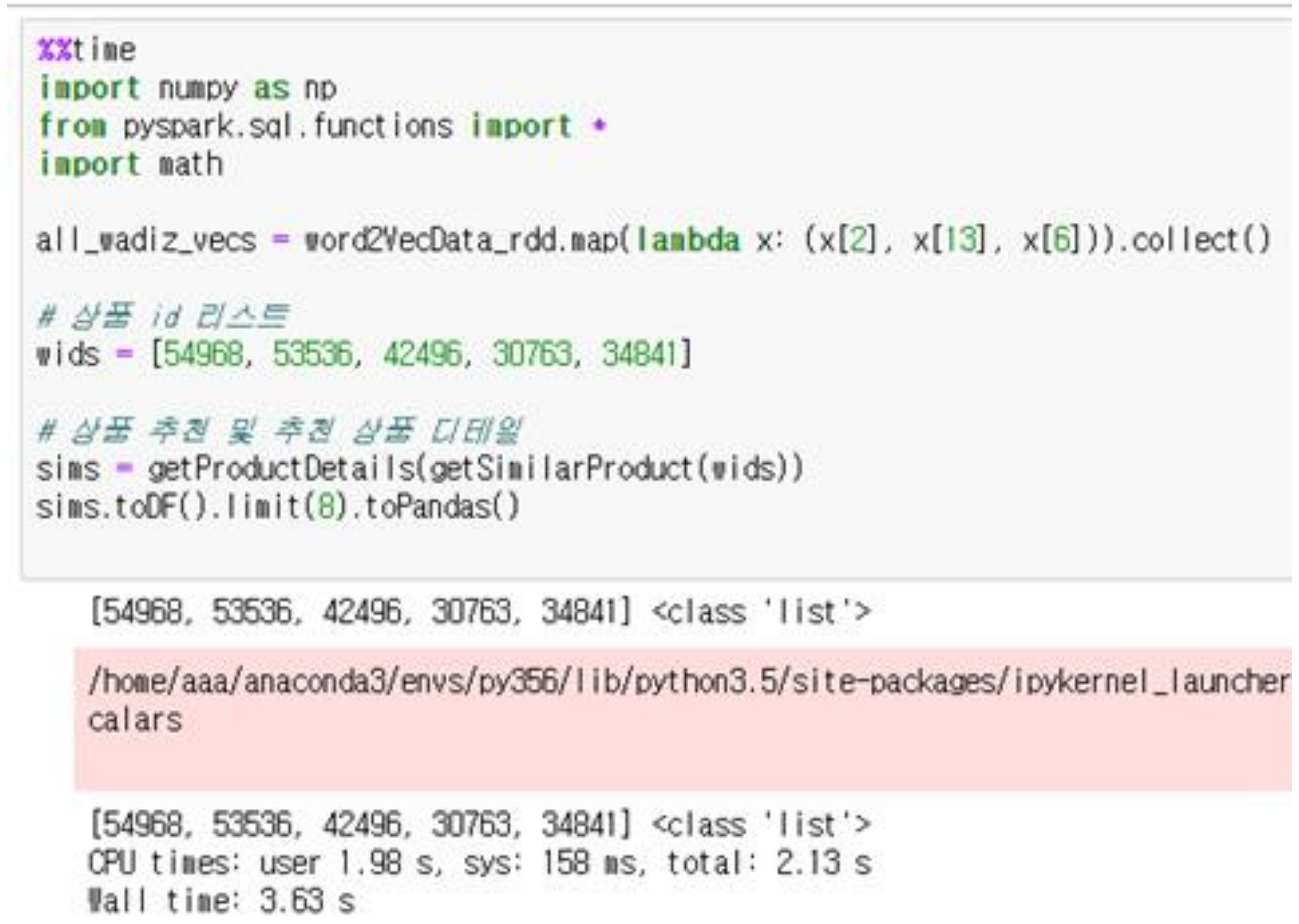
```
%%time
from pyspark.ml.feature import Tokenizer,HashingTF, Word2Vec

tokenizer = Tokenizer(inputCol='soop', outputCol='keywords')
wordData = tokenizer.transform(spark_df)
word2Vec = Word2Vec(vectorSize=100, minCount=5, inputCol='keywords', outputCol='word_vec', seed=123)
word2VecData = word2Vec.fit(wordData)
word2VecData = word2VecData.transform(wordData)
```

```
CPU times: user 7.3 ms, sys: 2.24 ms, total: 9.55 ms
Wall time: 3.43 s
```

Make word2Vector data

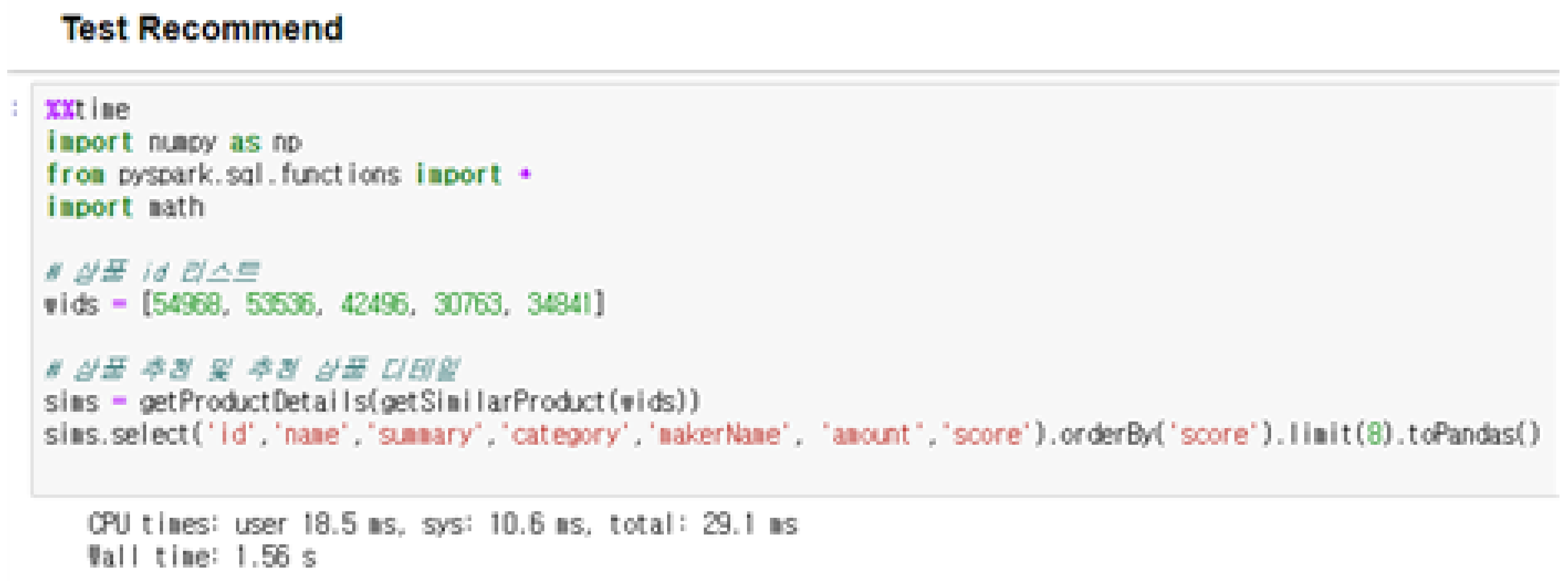
Data analysis - CBF

Bigdata project - Crowd funding analysis



```
%%time
import numpy as np
from pyspark.sql.functions import *
import math

all_wadiz_vecs = word2VecData_rdd.map(lambda x: (x[2], x[13], x[6])).collect()

# 상품 id 리스트
wids = [54968, 53536, 42496, 30763, 34841]

# 상품 추천 및 추천 상품 디테일
sims = getProductDetails(getSimilarProduct(wids))
sims.toDF().limit(8).toPandas()
```

```
[54968, 53536, 42496, 30763, 34841] <class 'list'>
/home/aaa/anaconda3/envs/py356/lib/python3.5/site-packages/ipykernel_launcher
calars

[54968, 53536, 42496, 30763, 34841] <class 'list'>
CPU times: user 1.98 s, sys: 158 ms, total: 2.13 s
Wall time: 3.63 s
```



Test Recommend

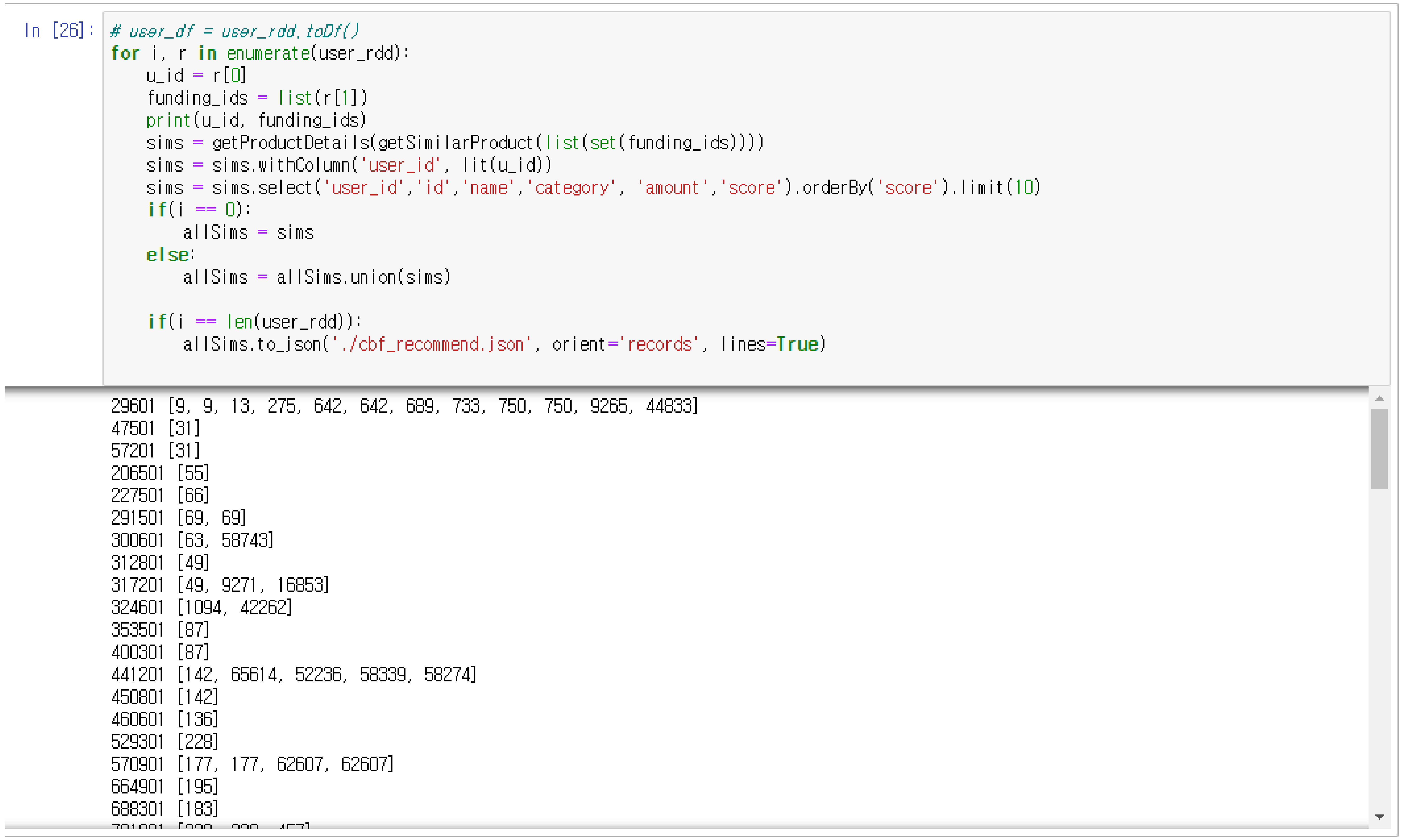
```
%%time
import numpy as np
from pyspark.sql.functions import *
import math

# 상품 id 리스트
wids = [54968, 53536, 42496, 30763, 34841]

# 상품 추천 및 추천 상품 디테일
sims = getProductDetails(getSimilarProduct(wids))
sims.select('id','name','summary','category','makerName', 'amount','score').orderBy('score').limit(8).toPandas()
```

```
CPU times: user 18.5 ms, sys: 10.6 ms, total: 29.1 ms
Wall time: 1.56 s
```

Make recommendation

Data analysis – CBF

Bigdata project - Crowd funding analysis



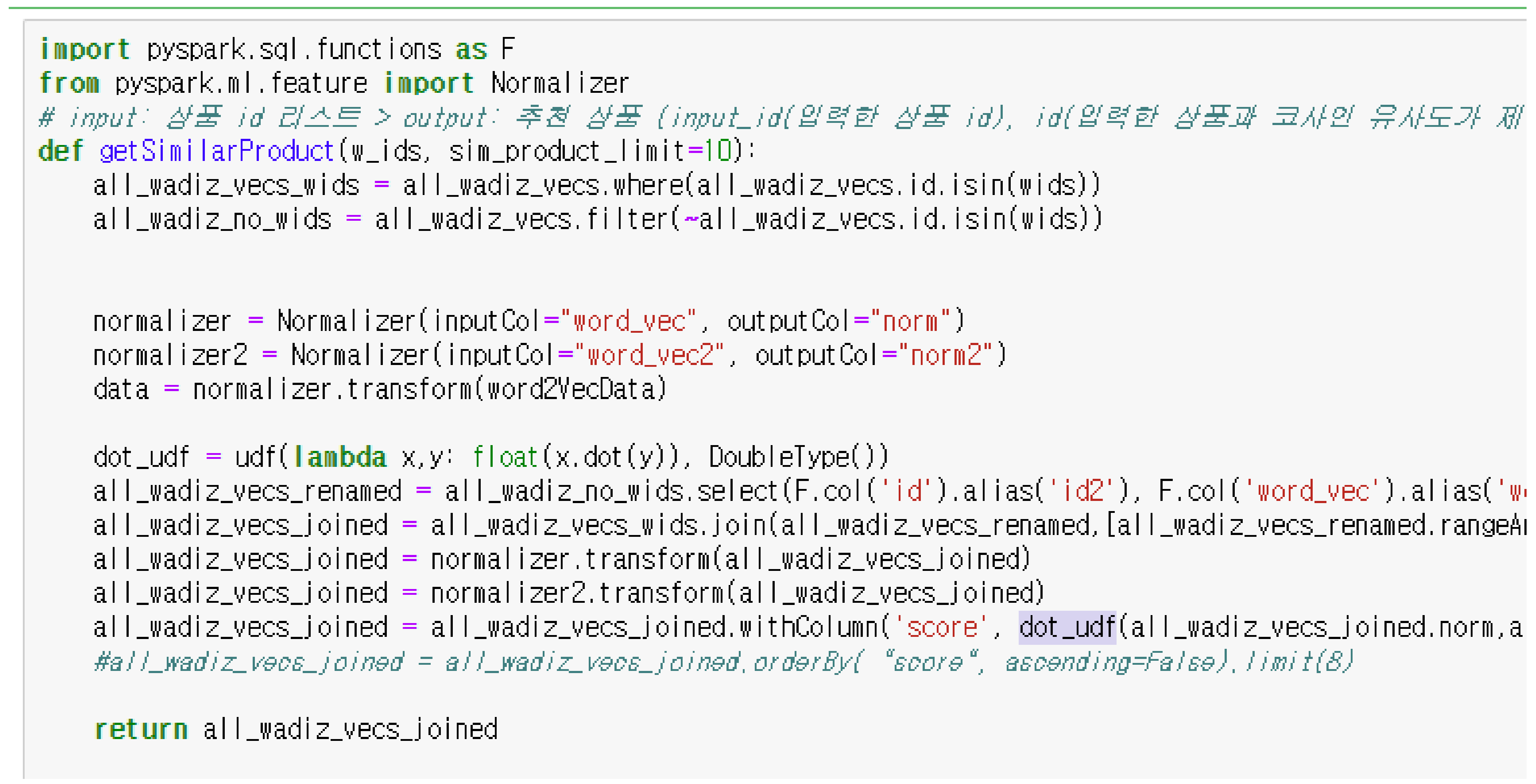
In [26]:
```
# user_df = user_rdd.toDf()
for i, r in enumerate(user_rdd):
    u_id = r[0]
    funding_ids = list(r[1])
    print(u_id, funding_ids)
    sims = getProductDetails(getSimilarProduct(list(set(funding_ids))))
    sims = sims.withColumn('user_id', lit(u_id))
    sims = sims.select('user_id','id','name','category', 'amount','score').orderBy('score').limit(10)
    if(i == 0):
        allSims = sims
    else:
        allSims = allSims.union(sims)

    if(i == len(user_rdd)):
        allSims.to_json('./cbf_recommend.json', orient='records', lines=True)
```

```
29601 [9, 9, 13, 275, 642, 642, 689, 733, 750, 750, 9265, 44833]
47501 [31]
57201 [31]
206501 [55]
227501 [66]
291501 [69, 69]
300601 [63, 58743]
312801 [49]
317201 [49, 9271, 16853]
324601 [1094, 42262]
353501 [87]
400301 [87]
441201 [142, 65614, 52236, 58339, 58274]
450801 [142]
460601 [136]
529301 [228]
570901 [177, 177, 62607, 62607]
664901 [195]
688301 [183]
```

Store json file

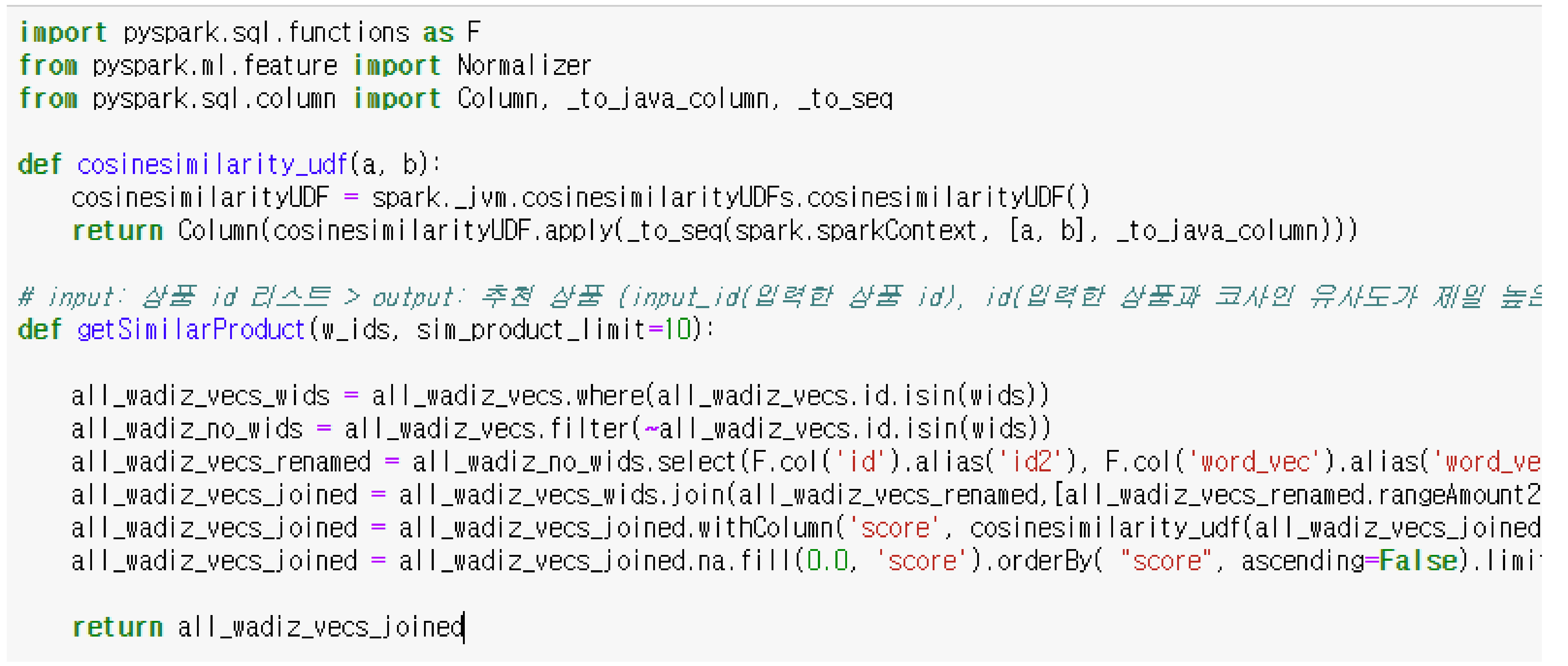
Data analysis - CBF

Bigdata project - Crowd funding analysis



```python
import pyspark.sql.functions as F
from pyspark.ml.feature import Normalizer
# input: 상품 id 리스트 > output: 추천 상품 (input_id(입력한 상품 id), id(입력한 상품과 코사인 유사도가 제
def getSimilarProduct(w_ids, sim_product_limit=10):
    all_wadiz_vecs_wids = all_wadiz_vecs.where(all_wadiz_vecs.id.isin(wids))
    all_wadiz_no_wids = all_wadiz_vecs.filter(~all_wadiz_vecs.id.isin(wids))


    normalizer = Normalizer(inputCol="word_vec", outputCol="norm")
    normalizer2 = Normalizer(inputCol="word_vec2", outputCol="norm2")
    data = normalizer.transform(word2VecData)

    dot_udf = udf(lambda x,y: float(x.dot(y)), DoubleType())
    all_wadiz_vecs_renamed = all_wadiz_no_wids.select(F.col('id').alias('id2'), F.col('word_vec').alias('w
    all_wadiz_vecs_joined = all_wadiz_vecs_wids.join(all_wadiz_vecs_renamed,[all_wadiz_vecs_renamed.rangeA
    all_wadiz_vecs_joined = normalizer.transform(all_wadiz_vecs_joined)
    all_wadiz_vecs_joined = normalizer2.transform(all_wadiz_vecs_joined)
    all_wadiz_vecs_joined = all_wadiz_vecs_joined.withColumn('score', dot_udf(all_wadiz_vecs_joined.norm,a
    #all_wadiz_vecs_joined = all_wadiz_vecs_joined.orderBy( "score", ascending=False).limit(8)

    return all_wadiz_vecs_joined
```

```
CPU times: user 22 ms, sys: 12.6 ms, total: 34.6 ms
Wall time: 4.65 s
```



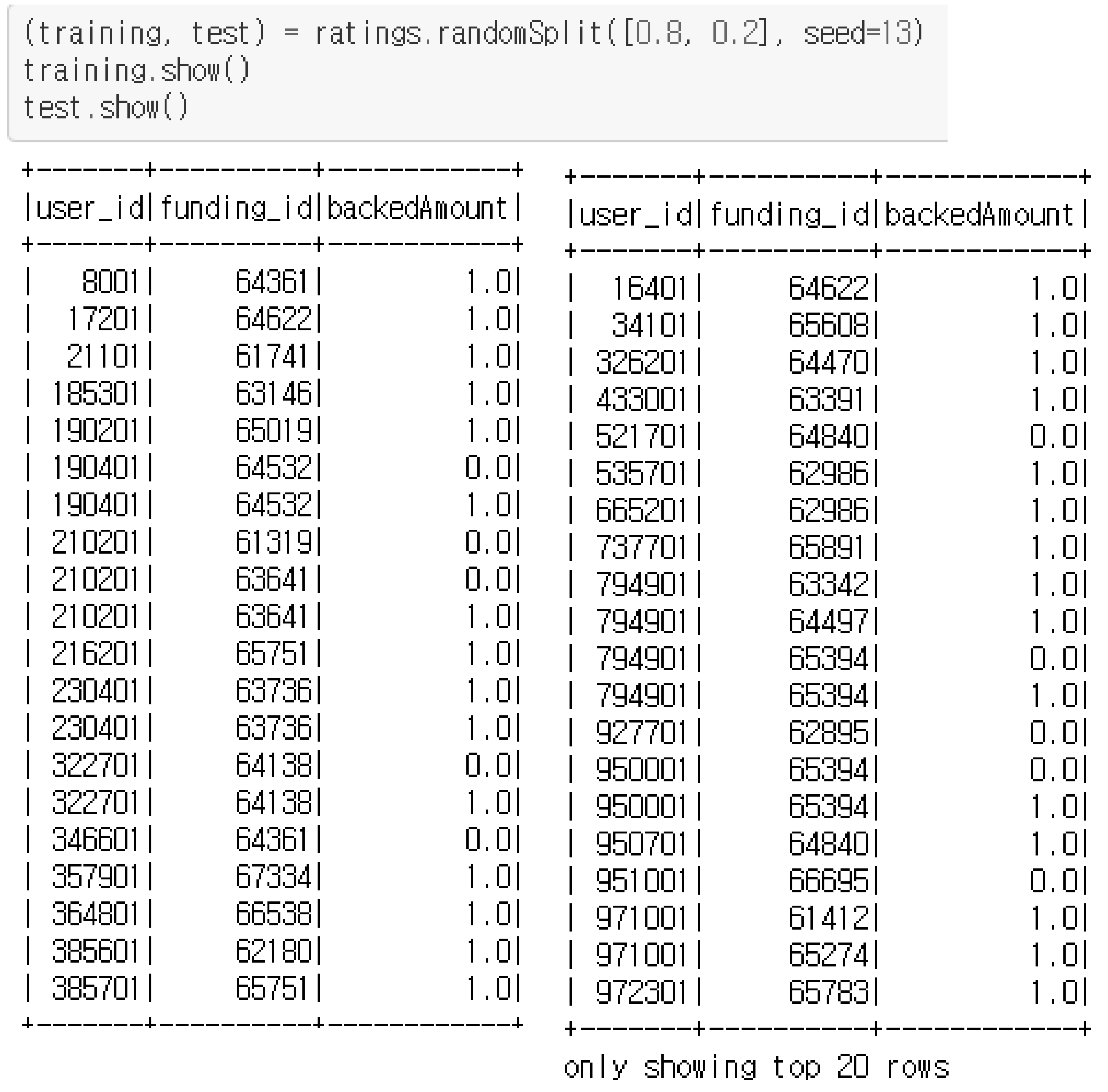
```python
import pyspark.sql.functions as F
from pyspark.ml.feature import Normalizer
from pyspark.sql.column import Column, _to_java_column, _to_seq

def cosinesimilarity_udf(a, b):
    cosinesimilarityUDF = spark._jvm.cosinesimilarityUDFs.cosinesimilarityUDF()
    return Column(cosinesimilarityUDF.apply(_to_seq(spark.sparkContext, [a, b], _to_java_column)))

# input: 상품 id 리스트 > output: 추천 상품 (input_id(입력한 상품 id), id(입력한 상품과 코사인 유사도가 제일 높
def getSimilarProduct(w_ids, sim_product_limit=10):

    all_wadiz_vecs_wids = all_wadiz_vecs.where(all_wadiz_vecs.id.isin(wids))
    all_wadiz_no_wids = all_wadiz_vecs.filter(~all_wadiz_vecs.id.isin(wids))
    all_wadiz_vecs_renamed = all_wadiz_no_wids.select(F.col('id').alias('id2'), F.col('word_vec').alias('word_ve
    all_wadiz_vecs_joined = all_wadiz_vecs_wids.join(all_wadiz_vecs_renamed,[all_wadiz_vecs_renamed.rangeAmount2
    all_wadiz_vecs_joined = all_wadiz_vecs_joined.withColumn('score', cosinesimilarity_udf(all_wadiz_vecs_joined
    all_wadiz_vecs_joined = all_wadiz_vecs_joined.na.fill(0.0, 'score').orderBy( "score", ascending=False).limi

    return all_wadiz_vecs_joined
```

```
CPU times: user 45.4 ms, sys: 2.14 ms, total: 47.5 ms
Wall time: 2.02 s
```

Optimization using Scala UDF

Data analysis - CF

Bigdata project - Crowd funding analysis



```
(training, test) = ratings.randomSplit([0.8, 0.2], seed=13)
training.show()
test.show()
```

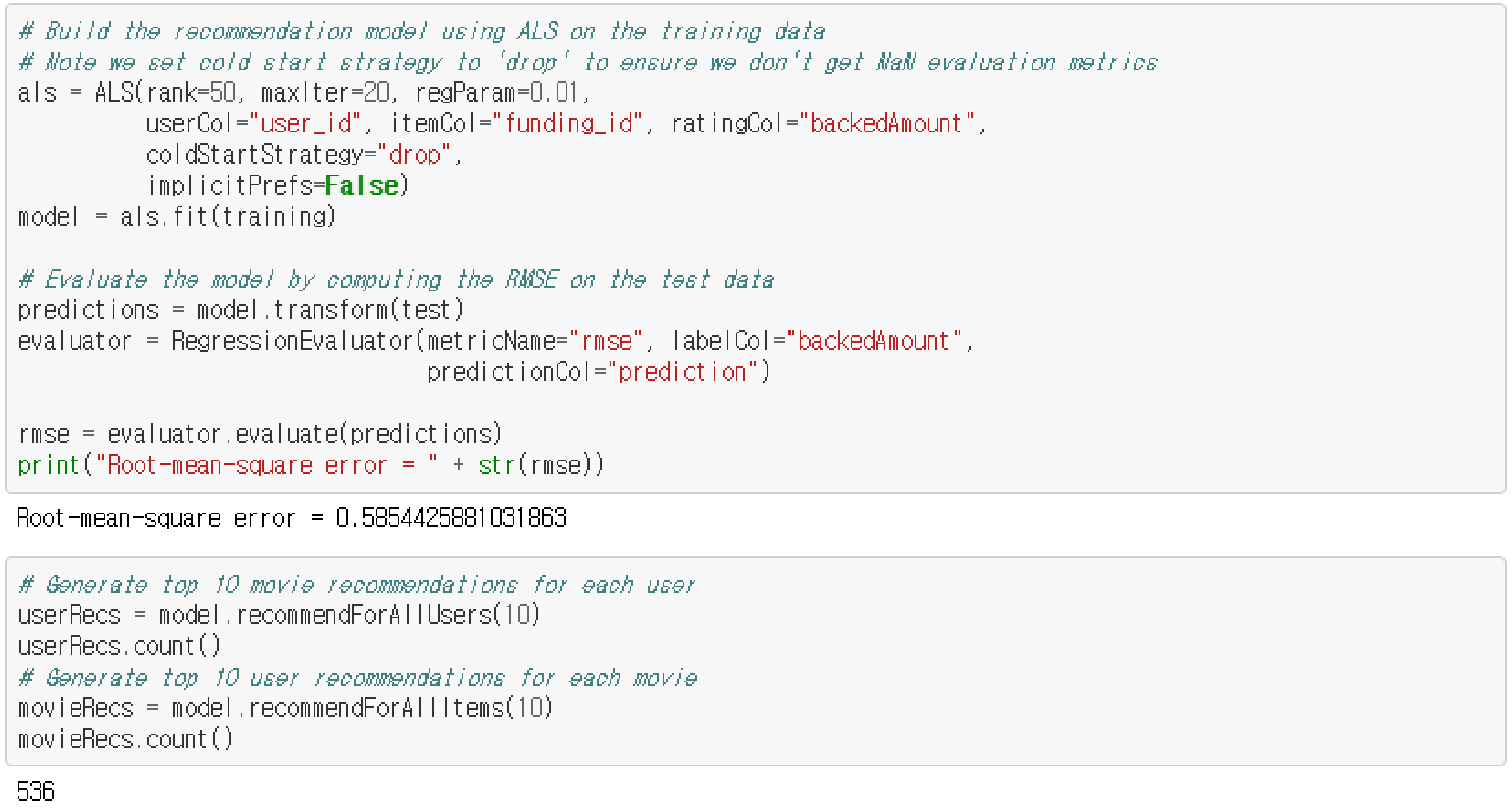
```
+-------+----------+------------+
|user_id|funding_id|backedAmount|
+-------+----------+------------+
|   8001|     64361|         1.0|
|  17201|     64622|         1.0|
|  21101|     61741|         1.0|
| 185301|     63146|         1.0|
| 190201|     65019|         1.0|
| 190401|     64532|         0.0|
| 190401|     64532|         1.0|
| 210201|     61319|         0.0|
| 210201|     63641|         0.0|
| 210201|     63641|         1.0|
| 216201|     65751|         1.0|
| 230401|     63736|         1.0|
| 230401|     63736|         1.0|
| 322701|     64138|         0.0|
| 322701|     64138|         1.0|
| 346601|     64361|         0.0|
| 357901|     67334|         1.0|
| 364801|     66538|         1.0|
| 385601|     62180|         1.0|
| 385701|     65751|         1.0|
+-------+----------+------------+
```

```
+-------+----------+------------+
|user_id|funding_id|backedAmount|
+-------+----------+------------+
|  16401|     64622|         1.0|
|  34101|     65608|         1.0|
| 326201|     64470|         1.0|
| 433001|     63391|         1.0|
| 521701|     64840|         0.0|
| 535701|     62986|         1.0|
| 665201|     62986|         1.0|
| 737701|     65891|         1.0|
| 794901|     63342|         1.0|
| 794901|     64497|         1.0|
| 794901|     65394|         0.0|
| 794901|     65394|         1.0|
| 927701|     62895|         0.0|
| 950001|     65394|         0.0|
| 950001|     65394|         1.0|
| 950701|     64840|         1.0|
| 951001|     66695|         0.0|
| 971001|     61412|         1.0|
| 971001|     65274|         1.0|
| 972301|     65783|         1.0|
+-------+----------+------------+
only showing top 20 rows
```

Making training set and test set for spark ALS

Data analysis - CF

Bigdata project - Crowd funding analysis



```
# Build the recommendation model using ALS on the training data
# Note we set cold start strategy to 'drop' to ensure we don't get NaN evaluation metrics
als = ALS(rank=50, maxIter=20, regParam=0.01,
          userCol="user_id", itemCol="funding_id", ratingCol="backedAmount",
          coldStartStrategy="drop",
          implicitPrefs=False)
model = als.fit(training)

# Evaluate the model by computing the RMSE on the test data
predictions = model.transform(test)
evaluator = RegressionEvaluator(metricName="rmse", labelCol="backedAmount",
                                predictionCol="prediction")

rmse = evaluator.evaluate(predictions)
print("Root-mean-square error = " + str(rmse))
```

```
Root-mean-square error = 0.5854425881031863
```

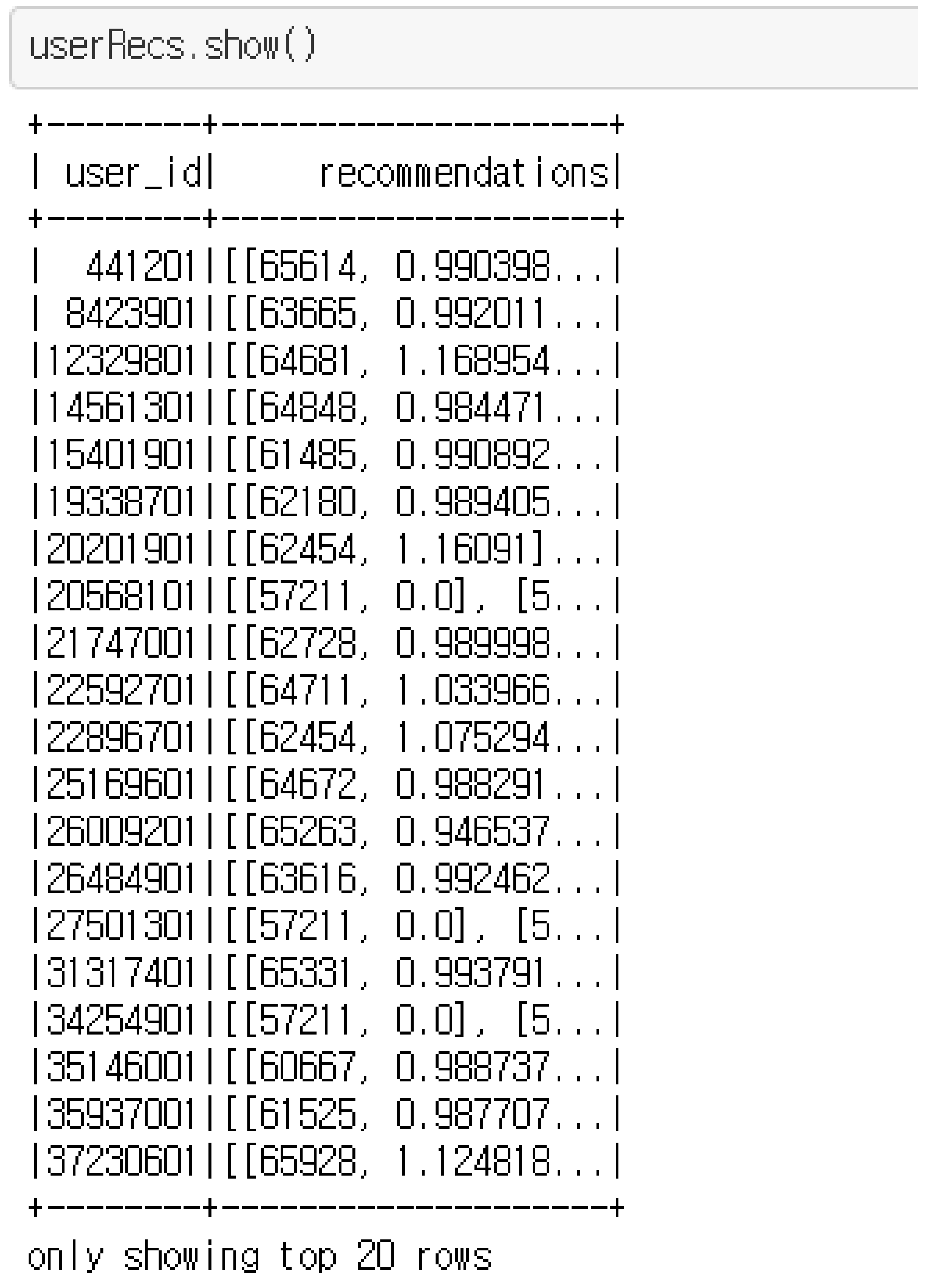
```
# Generate top 10 movie recommendations for each user
userRecs = model.recommendForAllUsers(10)
userRecs.count()
# Generate top 10 user recommendations for each movie
movieRecs = model.recommendForAllItems(10)
movieRecs.count()
```

```
536
```

Using spark ALS to analyze

Data analysis - CF

Bigdata project - Crowd funding analysis



```
userRecs.show()
```

```
+--------+--------------------+
| user_id|     recommendations|
+--------+--------------------+
|  441201|[[65614, 0.990398...|
| 8423901|[[63665, 0.992011...|
|12329801|[[64681, 1.168954...|
|14561301|[[64848, 0.984471...|
|15401901|[[61485, 0.990892...|
|19338701|[[62180, 0.989405...|
|20201901|[[62454, 1.16091]...|
|20568101|[[57211, 0.0], [5...|
|21747001|[[62728, 0.989998...|
|22592701|[[64711, 1.033966...|
|22896701|[[62454, 1.075294...|
|25169601|[[64672, 0.988291...|
|26009201|[[65263, 0.946537...|
|26484901|[[63616, 0.992462...|
|27501301|[[57211, 0.0], [5...|
|31317401|[[65331, 0.993791...|
|34254901|[[57211, 0.0], [5...|
|35146001|[[60667, 0.988737...|
|35937001|[[61525, 0.987707...|
|37230601|[[65928, 1.124818...|
+--------+--------------------+
only showing top 20 rows
```

Result of ALS !

Visualization

Bigdata project - Crowd funding analysis

Input user data

Input user id to recommend new product

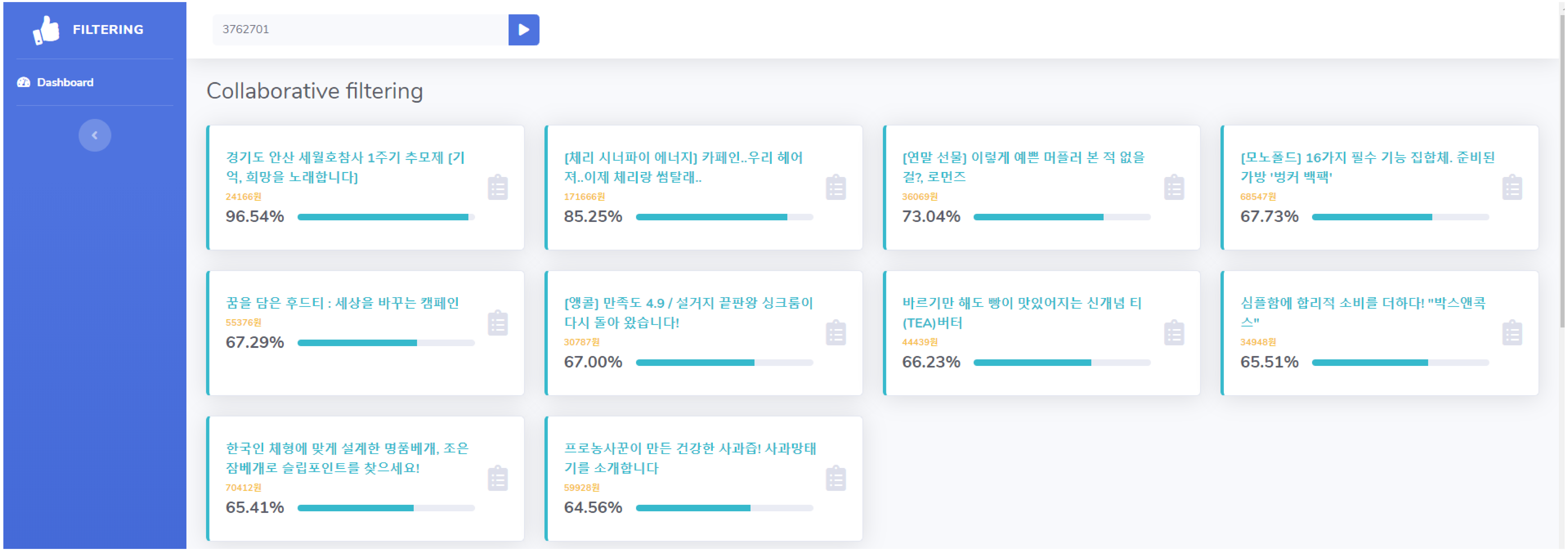
Our Web page

Show recommend

Show chart and graph of result and comparison two recommend algorithm

Visualization

Bigdata project - Crowd funding analysis



3762701

Collaborative filtering

경기도 안산 세월호참사 1주기 추모제 [기억, 희망을 노래합니다]
24166원
96.54%

[체리 시너파이 에너지] 카페인..우리 헤어져..이제 체리랑 썸탈래..
171666원
85.25%

[연말 선물] 이렇게 예쁜 머플러 본 적 없을걸?, 로먼즈
36069원
73.04%

[모노폴드] 16가지 필수 기능 집합체. 준비된 가방 '벙커 백팩'
68547원
67.73%

꿈을 담은 후드티 : 세상을 바꾸는 캠페인
55376원
67.29%

[앵콜] 만족도 4.9 / 설거지 끝판왕 싱크룸이 다시 돌아 왔습니다!
30787원
67.00%

바르기만 해도 빵이 맛있어지는 신개념 티(TEA)버터
44439원
66.23%

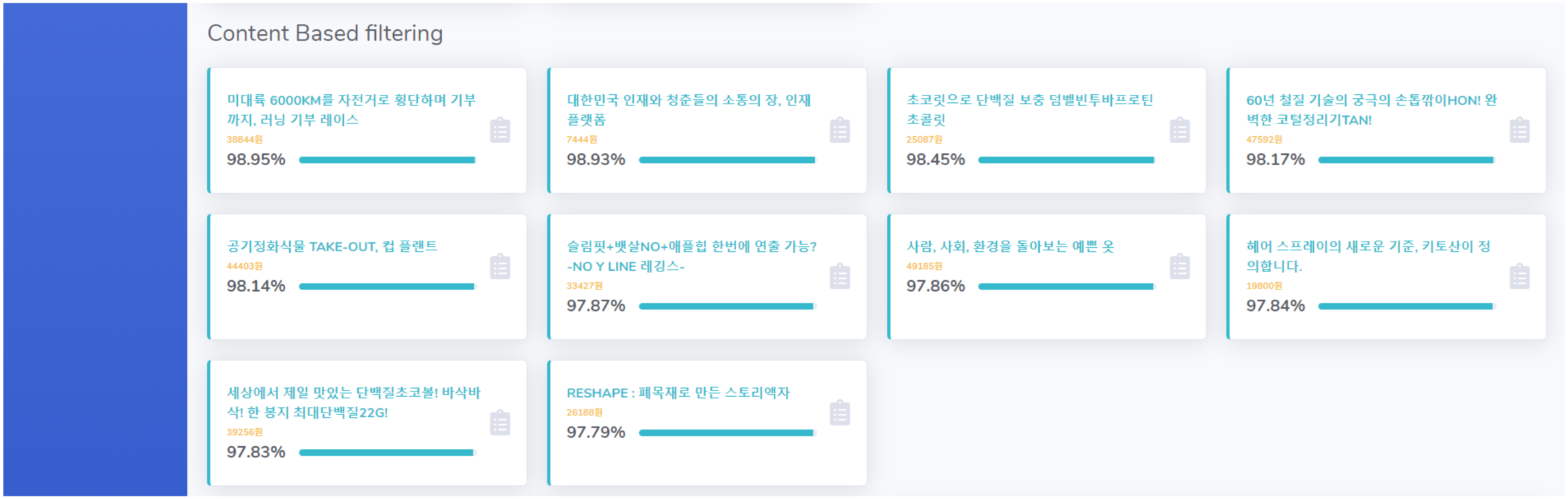
심플함에 합리적 소비를 더하다! "박스앤콕스"
34948원
65.51%

한국인 체형에 맞게 설계한 명품베개, 조은잠베개로 슬립포인트를 찾으세요!
70412원
65.41%

프로농사꾼이 만든 건강한 사과즙! 사과망태기를 소개합니다
59928원
64.56%

Visualization

Bigdata project - Crowd funding analysis



Content Based filtering

미대륙 6000KM를 자전거로 횡단하며 기부까지, 러닝 기부 레이스
38844원
98.95%

대한민국 인재와 청춘들의 소통의 장, 인재플랫폼
7444원
98.93%

초코릿으로 단백질 보충 덤벨빈투바프로틴초콜릿
25087원
98.45%

60년 철질 기술의 궁극의 손톱깎이HON! 완벽한 코털정리기TAN!
47592원
98.17%

공기정화식물 TAKE-OUT, 컵 플랜트
44403원
98.14%

슬림핏+뱃살NO+애플힙 한번에 연출 가능? -NO Y LINE 레깅스-
33427원
97.87%

사람, 사회, 환경을 돌아보는 예쁜 옷
49185원
97.86%

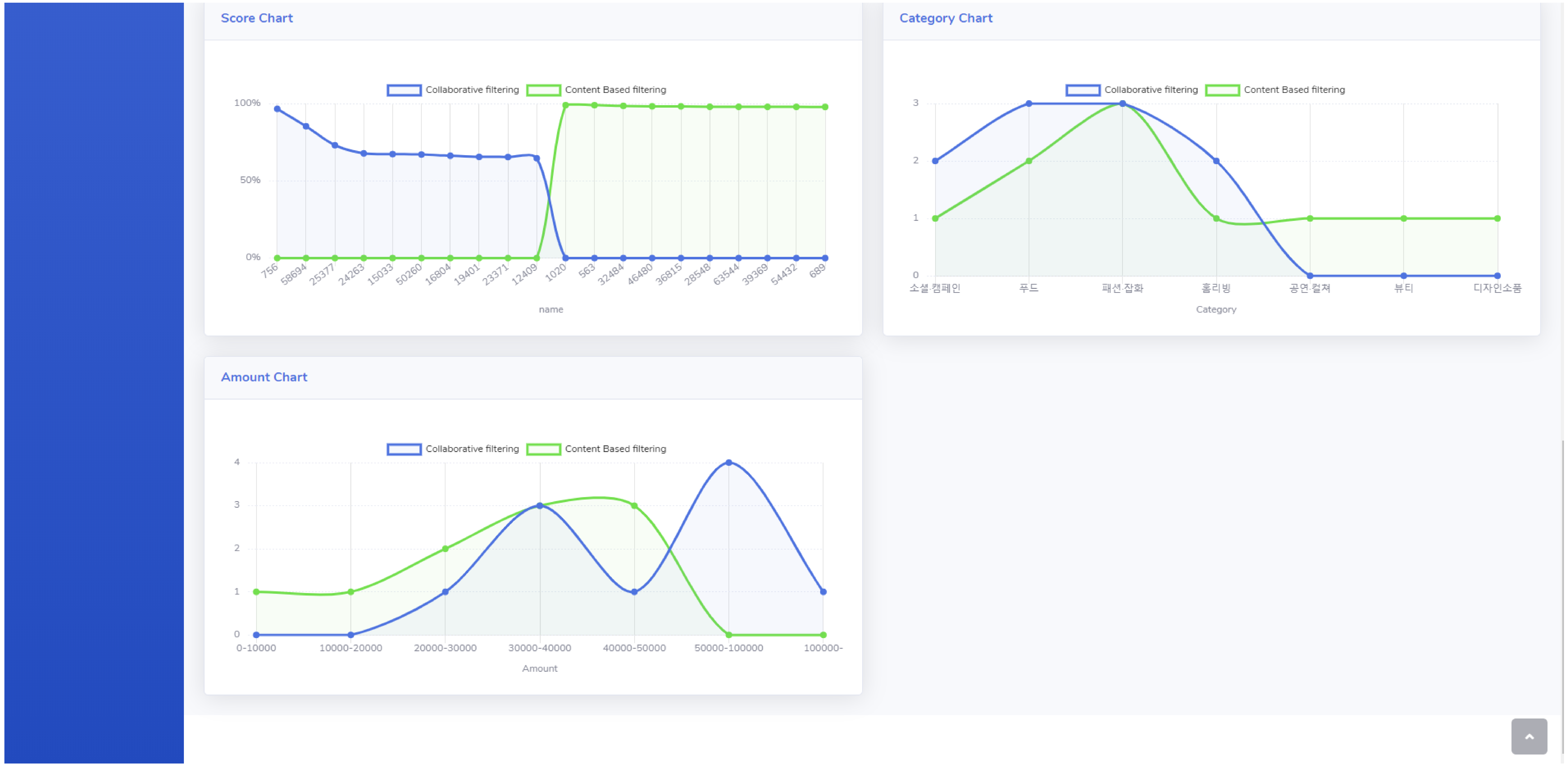
헤어 스프레이의 새로운 기준, 키토산이 정의합니다.
19800원
97.84%

세상에서 제일 맛있는 단백질초코볼! 바삭바삭! 한 봉지 최대단백질22G!
39256원
97.83%

RESHAPE : 폐목재로 만든 스토리액자
26188원
97.79%

Visualization

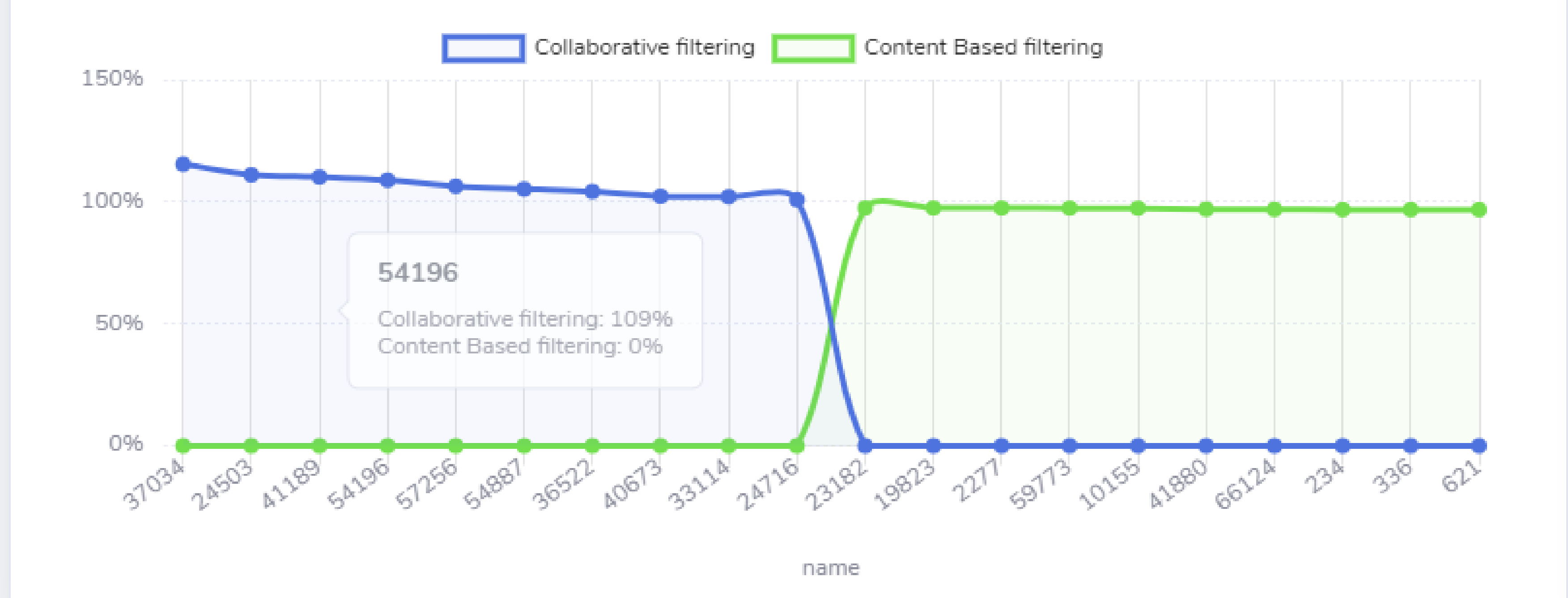
Bigdata project - Crowd funding analysis



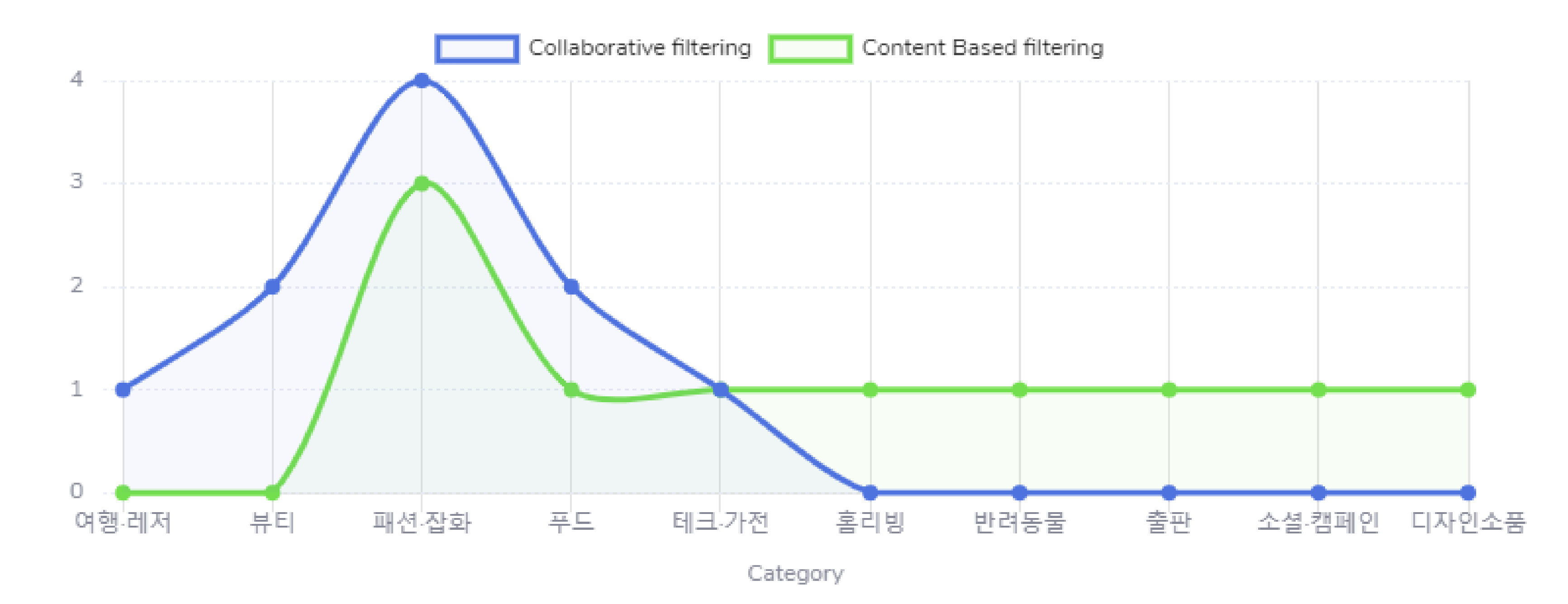
Visualization

Bigdata project - Crowd funding analysis

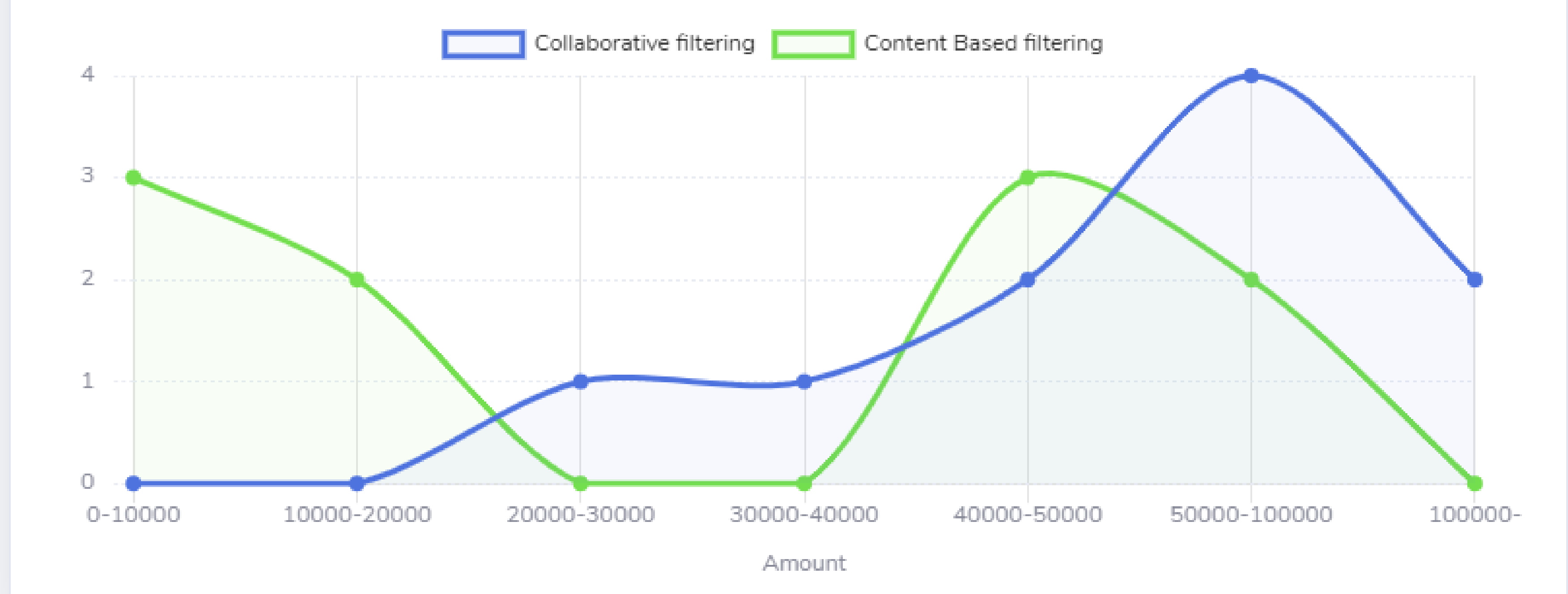
Score Chart



Category Chart



Amount Chart



Result

Bigdata project - Crowd funding analysis

Kind of product

The results of two algorithm were completely different

Product category

When there were many items funded by a specific user, the results of the funding category recommending of the two algorithms were similar.

Price amount

In CBF, we can show similar price group.
But, in CF's recommending was didn't to be much related.

Individual role

Bigdata project - Crowd funding analysis

김성연
Project leader

Scrum master

Data acquisition

Data analysis

박기범
member

Data acquisition

Make PPT and presentation

박우진
member

Data acquisition

Data analysis

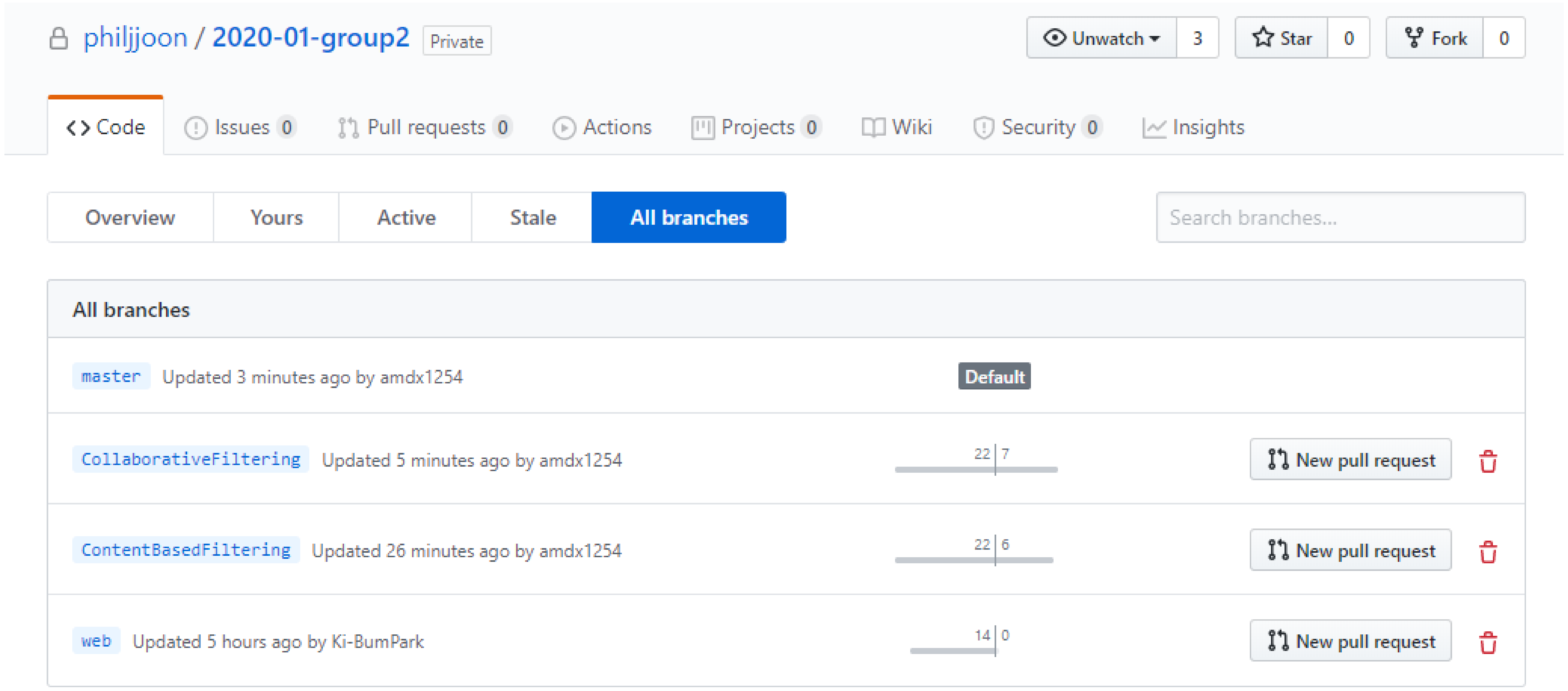
홍성현
member

Data acquisition

Data visualization

Individual role

Bigdata project - Crowd funding analysis



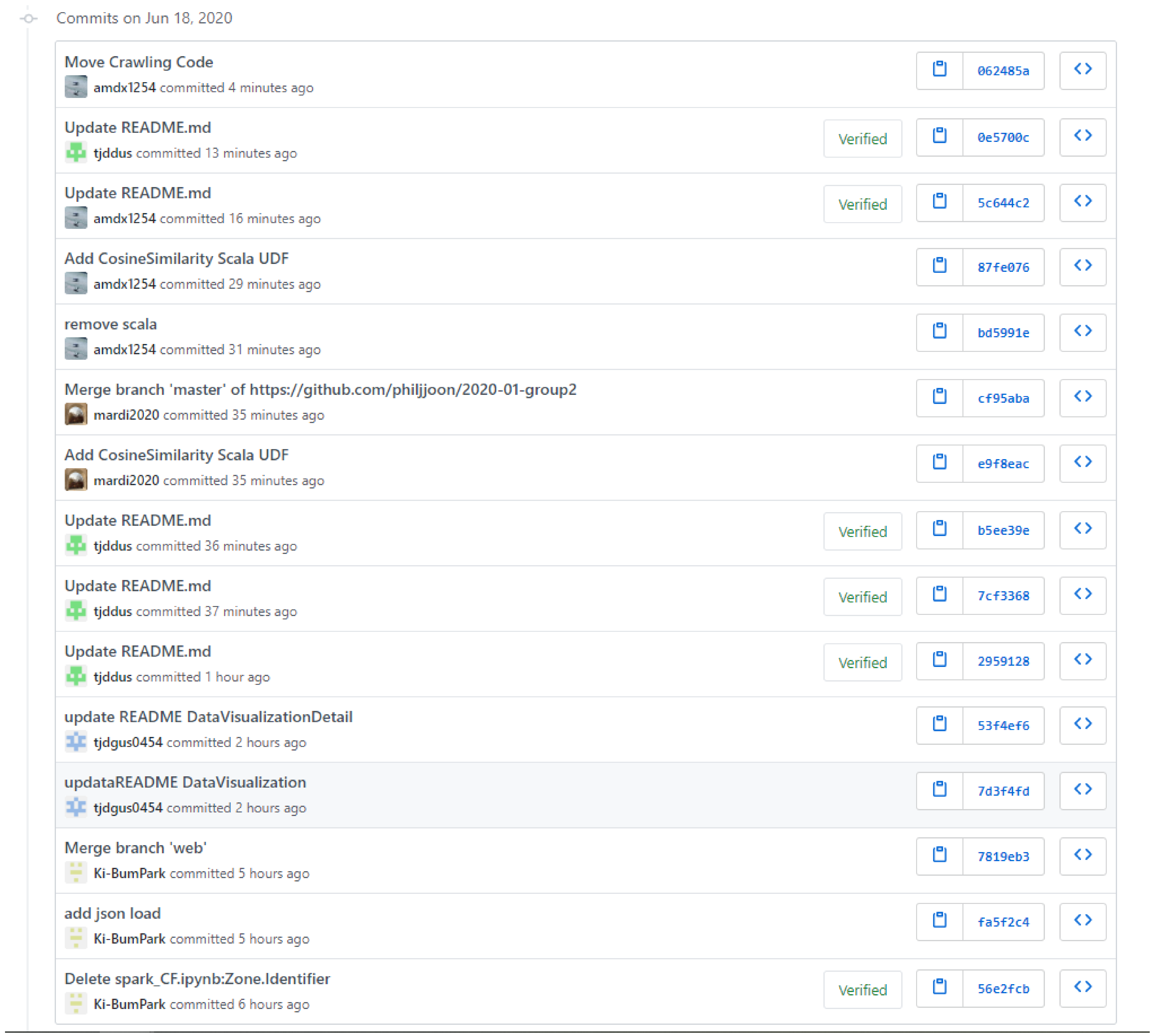
philjjoon / 2020-01-group2 Private

All branches

master	Updated 3 minutes ago by amdx1254	Default	
CollaborativeFiltering	Updated 5 minutes ago by amdx1254	22 | 7	New pull request
ContentBasedFiltering	Updated 26 minutes ago by amdx1254	22 | 6	New pull request
web	Updated 5 hours ago by Ki-BumPark	14 | 0	New pull request

Individual role

Bigdata project - Crowd funding analysis



Commits on Jun 18, 2020

Move Crawling Code
amdx1254 committed 4 minutes ago — 062485a

Update README.md
tjddus committed 13 minutes ago — Verified — 0e5700c

Update README.md
amdx1254 committed 16 minutes ago — Verified — 5c644c2

Add CosineSimilarity Scala UDF
amdx1254 committed 29 minutes ago — 87fe076

remove scala
amdx1254 committed 31 minutes ago — bd5991e

Merge branch 'master' of https://github.com/philjjoon/2020-01-group2
mardi2020 committed 35 minutes ago — cf95aba

Add CosineSimilarity Scala UDF
mardi2020 committed 35 minutes ago — e9f8eac

Update README.md
tjddus committed 36 minutes ago — Verified — b5ee39e

Update README.md
tjddus committed 37 minutes ago — Verified — 7cf3368

Update README.md
tjddus committed 1 hour ago — Verified — 2959128

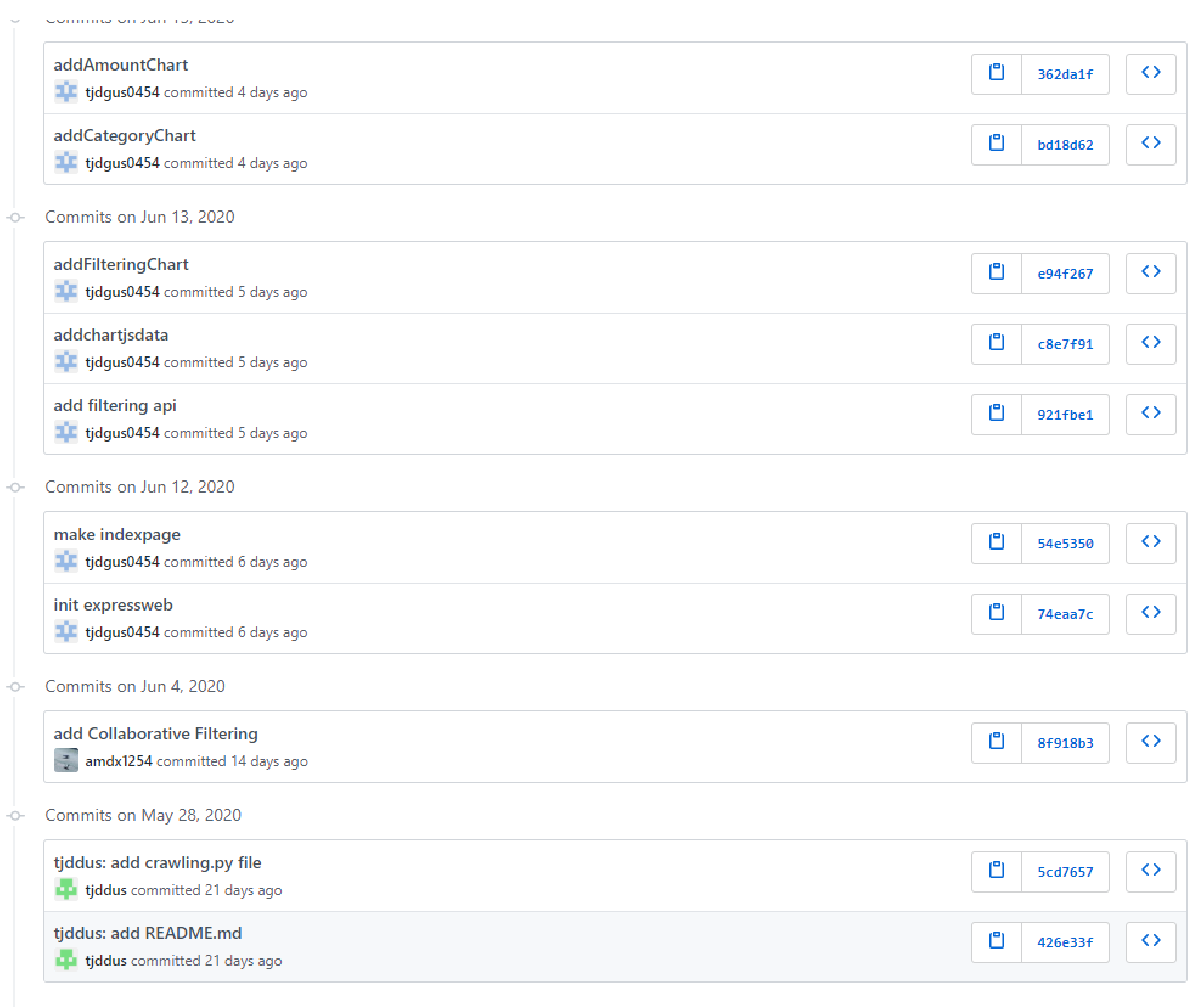
update README DataVisualizationDetail
tjdgus0454 committed 2 hours ago — 53f4ef6

updataREADME DataVisualization
tjdgus0454 committed 2 hours ago — 7d3f4fd

Merge branch 'web'
Ki-BumPark committed 5 hours ago — 7819eb3

add json load
Ki-BumPark committed 5 hours ago — fa5f2c4

Delete spark_CF.ipynb:Zone.Identifier
Ki-BumPark committed 6 hours ago — Verified — 56e2fcb



addAmountChart
tjdgus0454 committed 4 days ago — 362da1f

addCategoryChart
tjdgus0454 committed 4 days ago — bd18d62

Commits on Jun 13, 2020

addFilteringChart
tjdgus0454 committed 5 days ago — e94f267

addchartjsdata
tjdgus0454 committed 5 days ago — c8e7f91

add filtering api
tjdgus0454 committed 5 days ago — 921fbe1

Commits on Jun 12, 2020

make indexpage
tjdgus0454 committed 6 days ago — 54e5350

init expressweb
tjdgus0454 committed 6 days ago — 74eaa7c

Commits on Jun 4, 2020

add Collaborative Filtering
amdx1254 committed 14 days ago — 8f918b3

Commits on May 28, 2020

tjddus: add crawling.py file
tjddus committed 21 days ago — 5cd7657

tjddus: add README.md
tjddus committed 21 days ago — 426e33f

Thank you!

See You Next Time