



주류 추천 어플리케이션

# DRUNKEN WHALE

술 고 래

TAVE CONFERENCE

알만추  
민지웅 박명진 안유진 홍서영 김동현



### 개발 언어

- JAVA, Python, R

### 개발 환경

- Android Studio, Jupyter Notebook, R studio

### 데이터베이스

- Firebase

### 라이브러리

- Python: NumPy, Pandas, SciPy, Scikit-Learn
- R: readr, proxy, dplyr



### 기본 컨셉

- ‘술’에 대한 정보 제공 및 추천

### 1. 술 검색 및 정보 기능

- 술 검색
- 원하는 술에 대한 상세 정보 제공
- 좋아하는 술 즐겨찾기

### 2. 술 추천 기능

사용자가 선택한 술과 유사한 술을 추천

- 전체 주종 내에서 추천 → 와인이면 비슷한 풍미를 가진 막걸리, 맥주 추천
- 선택한 술과 같은 주종 내에서만 추천 → 와인이면 비슷한 와인 추천



크게 두가지 접근방법이 존재

### 1. Collaborative Filtering

- 사용자의 평가 내역을 분석하여 추천
- 좋은 성능, 사용자의 행동 패턴에 따라 적절한 추천
- 그러나 수집된 정보의 양이 많아야 좋은 결과가 나온다.

### 2. Content-based

- 아이템의 내용을 분석하여 추천
- 적은 정보만으로도 좋은 추천 가능
- 그러나 모델링 방식에 따라 정확도가 많이 달라지고, 비슷한 아이템끼리만 추천이 가능하여 추천 범위가 제한된다.



### 1. Collaborative Filtering

- User-based

유저의 행위를 측정 및 분석하고 이를 기반으로 유저간 유사도 측정하여 유사한 유저끼리 해당 유저의 선호 아이템 기반으로 추천

- Item-based

아이템간의 유사도를 측정하며, 유저가 어떤 아이템을 선호하면 유사한 다른 아이템을 추천

### 2. Content-based

- 아이템이나 유저를 분석하여 비슷한 아이템을 추천



### Content-based (콘텐츠 기반) 추천 시스템 사용 이유

- **유저 데이터가 없음**  
: User-based를 쓰기 위해서는 주류 구매자에 대한 정보 및 선호하는 주류에 대한 정보가 있어야함
- **아이템 선호도에 대한 정보가 없음**  
: Item-based를 쓰기 위해서는 그 아이템 선호도에 대한 정보가 있어야함  
  
→ Collaborative Filtering 사용 불가능
- Content-based는 아이템과 유저간의 액션을 분석하는 것이 아니라 **콘텐츠 자체를 분석!**  
→ 사용 가능



### Netflix의 추천 콘텐츠 시스템 작동 방법

#### 아이템 정보 Data

- 넷플릭스 서비스와의 상호작용 - 시청 기록, 다른 콘텐츠 평가 결과 등
- 유사한 취향을 가진 회원 및 넷플릭스 서비스에서의 선호 대상
- 장르, 카테고리, 배우, 출시연도 등 콘텐츠 관련 정보

#### 사용자 정보 Data

- 하루 중 시청 시간대
- 넷플릭스를 시청하는 디바이스
- 시청 시간



## 데이터 설명

---

<u>id</u>	<u>class</u>	<u>category</u>	<u>name</u>	<u>percent</u>	<u>origin</u>	<u>producer</u>	<u>wine_grape</u>	<u>whisky_category</u>	<u>sweet</u>	<u>light</u>	<u>soft</u>	<u>bitter</u>	<u>clean</u>	<u>smell</u>
10	wine	와인-레드	볼베르	15.5	스페인	보데가스 볼베르	템프라니오		1	1	1	0	0	0
85	soju	소주-	처음처럼	17.5		롯데칠성음료			0	1	0	1	0	1
237	위스키	위스키-American	잭다니엘 허니 700ml	35	미국	잭다니엘		Tennessee	0	0	0	0	0	0
597	beer	맥주-밀맥주	1664 블랑	5	폴란드	CARLSBERG SUPPLY COMPANY			1	0	0	0	1	0

총 656개의 술 데이터



## 데이터 설명

id	class	category	name	percent	origin	producer	wine_grape	whisky_category	sweet	light	soft	bitter	clean	smell
				도수	원산지	제조사	포도 품종 (와인)					- 풍미 6가지 요소 -		

### class 대분류

- wine / makgeoli / beer / vodka / soju / whisky / korean (전통주)
- 총 7개의 대분류

### category 중분류

- 와인-레드 / 와인-화이트 / 위스키-American / 위스キー-Irish / 맥주-IPA / 맥주-라거 등
- 총 24개의 중분류



### 1) Cosine Similarity

#### 코사인 유사도

내적공간의 두 벡터간 각도의 코사인값을 이용하여 측정된 벡터간의 유사한 정도

##### 특징

1. -1은 서로 완전히 반대되는 경우를 의미
2. 0은 서로 독립적인 경우를 의미
3. 1은 서로 완전히 같은 경우를 의미

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$



## 1) Cosine Similarity

### 코사인 데이터 계산 예시 - 가상 데이터

예시) 2020년도 수능 등급컷 예상점수 및 실제 수능 등급컷

국어영역	1등급	2등급	3등급	4등급	5등급	6등급	7등급	8등급
수능	91	85	77	67	55	43	32	23
유웨이	91	84	76	65	55	43	26	20
진학사	91	85	77	66	55	42	32	22

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

코사인 거리 =  $1 - \text{similarity}$

$$\text{수능과 유웨이 코사인 유사도} = \frac{\sum(91 * 91 + 85 * 84 + \dots + 32 * 26 + 23 * 20)}{\sqrt{91^2 + 85^2 + \dots + 32^2 + 23^2} * \sqrt{91^2 + 84^2 + \dots + 26^2 + 20^2}}$$

	코사인 유사도	코사인거리
수능&유웨이	0.999338	0.000661973
수능&진학사	0.999962	0.000038227

국어영역에 대해 2020년도 수능등급 컷과 유사한 곳은 진학사



## 1) Cosine Similarity

## 코사인 데이터 계산 예시 - 실제 데이터

	sweet	light	soft	bitter	clean	smell
도멘드 벨렌, 부르고뉴 메종 디유	1	1	0	1	1	0
석로주	0	1	0	1	1	0
클라랑스 딜롱, 클라랑델 루즈	0	0	1	1	0	1
배다리쌀막걸리	1	1	1	0	1	0

## 코사인 거리

	도멘드벨렌	석로주	클라랑스	배다리쌀막걸리
도멘드벨伦	0.000	0.134	0.711	0.250
석로주	0.134	0.000	0.667	0.423
클라랑스	0.711	0.667	0.000	0.711
배다리쌀막걸리	0.250	0.423	0.711	0.000

도멘드 벨렌과 가장 유사한 술은 석로주

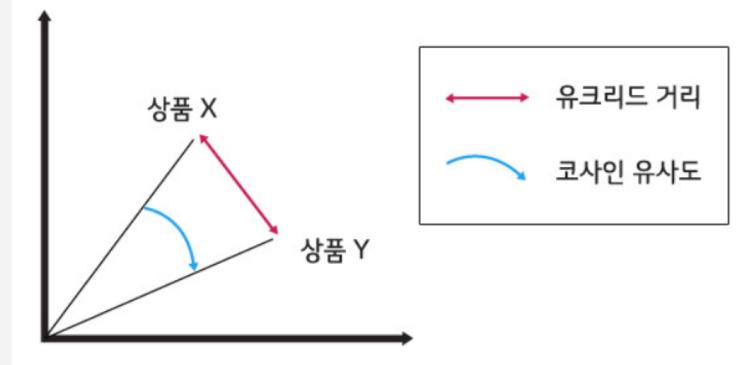


### 2) Euclidean Distance

#### 유클리드 거리

n차원의 공간에서 두 점간의 거리를 알아내는 공식

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$





### 1) 서영 - Euclidean Distance + Cosine Similarity

도수 유사도는 **euclidean distance**로 구했고, 풍미 및 기타 정보의 유사도는 **cosine similarity**로 구했다.

입력된 술의 고유 id와 비슷한 술의 고유 id를 return하는 프로그램

```
def general_recommendation():
```

도수 유사도 50% 풍미(6가지) 유사도 50% 고려해 주종 상관 없이 비슷한 술을 찾아줌

```
def wine_recommendation():
```

도수 유사도 50% 기타(category, origin, producer, wine\_grape, flavour) 50% 고려해 같은 대분류 내에서 비슷한 와인을 찾아줌



## 1) 서영 - Euclidean Distance + Cosine Similarity

```
[ ] # 도수 유사도 구하기
percent = data['percent']
dist_pair = []

# y축에 임의로 0을 부여한 거리 순서쌍 생성
for i in range(0, len(percent)):
    temp = []
    temp.append(data.loc[i]['percent'])
    temp.append(0)
    dist_pair.append(temp)

# get a distance matrix
df = pd.DataFrame(dist_pair, columns=['x', 'y'])
dist_matrix = distance_matrix(df.values, df.values)

# 정규화
min_max_scaler = MinMaxScaler()
regularised = min_max_scaler.fit_transform(dist_matrix)

# 1에서 빼줘서 더 가까운 것이 우선순위를 갖도록 변경하기
one_matrix = np.ones((654, 654))

final_dist = one_matrix - regularised

# 확인
print(final_dist)
```

```
[[1.          0.97560976 0.98795181 ... 0.82352941 0.82352941 0.82352941]
 [0.97619048 1.          0.98795181 ... 0.85154062 0.85154062 0.85154062]
 [0.98809524 0.98780488 1.          ... 0.83753501 0.83753501 0.83753501]
 ...
 [0.85       0.87073171 0.86024096 ... 1.          1.          1.          ]
 [0.85       0.87073171 0.86024096 ... 1.          1.          1.          ]
 [0.85       0.87073171 0.86024096 ... 1.          1.          1.          ]]
```

```
[ ] # flavour word list 만들기
flavour_list = [] # empty list

for i in range(0, len(data)):
    temp = ""

    if data.loc[i]['sweet'] == 1:
        temp = temp + "sweet "
    if data.loc[i]['light'] == 1:
        temp = temp + "light "
    if data.loc[i]['soft'] == 1:
        temp = temp + "soft "
    if data.loc[i]['bitter'] == 1:
        temp = temp + "bitter "
    if data.loc[i]['clean'] == 1:
        temp = temp + "clean "
    if data.loc[i]['smell'] == 1:
        temp = temp + "smell "

    flavour_list.append(temp)

# 해당 리스트 데이터 프레임에 추가
data["flavour"] = pd.DataFrame({"flavour":flavour_list})
flavour = data['flavour']

# flavour 코사인 유사도 구하기
# instantiating and generating the count matrix
count = CountVectorizer()
count_matrix = count.fit_transform(flavour)

# generating the cosine similarity matrix
cosine_sim = cosine_similarity(count_matrix, count_matrix)

# 확인
print(cosine_sim)
```

```
E [[1.          0.5          0.35355339 ... 0.57735027 0.5          0.70710678]
 [0.5          1.          0.70710678 ... 0.57735027 1.          0.          ]
 [0.35355339 0.70710678 1.          ... 0.81649658 0.70710678 0.          ]
 ...
 [0.57735027 0.57735027 0.81649658 ... 1.          0.57735027 0.          ]
 [0.5          1.          0.70710678 ... 0.57735027 1.          0.          ]
 [0.70710678 0.          0.          ... 0.          0.          1.          ]]
```

## 각자 구현한 코드

텍스트의 코사인 유사도를 구하기 위해  
변수를 텍스트화



## 각자 구현한 코드

### 1) 서영 - Euclidean Distance + Cosine Similarity

#### BOW(bag of words)

주로 자연어 처리에서 사용되는 방법으로 문장을 단어의 빈도수로 나타낸다.

이렇게 해서 만들어진 매트릭스로 코사인 유사도를 구함

'와인-레드 프랑스 도멘드벨렌 피노누아 sweet light bitter clean ',  
'와인-레드 칠레 몬테스 피노누아 bitter ' ,

Index	와인-레드	프랑스	칠레	도멘드벨렌	몬테스	피노누아	sweet	light	soft	bitter	clean	smell
1	1	1	0	1	0	1	1	1	0	1	1	0
2	1	0	1	0	1	1	0	0	0	1	0	0



## 각자 구현한 코드

### 1) 서영 - Euclidean Distance + Cosine Similarity

```
[ ] # 도수, flavour를 모두 고려한 similarity 구하기 (weight는 각각 0.5)
new_sim = 0.5 * cosine_sim + 0.5 * final_dist

print(new_sim)

[ [1.          0.73780488 0.6707526   ... 0.70043984 0.66176471 0.7653181  ],
  [0.73809524 1.          0.84752929 ... 0.71444544 0.92577031 0.42577031],
  [0.67082431 0.84745583 1.          ... 0.8270158  0.7723209  0.41876751],
  ...
  [0.71367513 0.72404099 0.83836877 ... 1.          0.78867513 0.5       ],
  [0.675       0.93536585 0.78367387 ... 0.78867513 1.          0.5       ],
  [0.77855339 0.43536585 0.43012048 ... 0.5       0.5       1.       ]]
```



## 각자 구현한 코드

### 1) 서영 - Euclidean Distance + Cosine Similarity

```
[ ] # 여러 주중 내 추천
# 주중에 상관 없이 도수와 풍미만 고려해 비슷한 술을 추천해줌
# 항목: 도수, 풍미 (약 6개 항목)

# 고유 id를 넣으면 해당 술과 비슷한 Top 10의 id를 return
def general_recommendation(input_id, new_sim = new_sim):

    # 고유 id로 index 찾기
    idx = data.index[data['id'] == input_id].tolist() # Int64Index 형식이라 list로 바꾸어줌

    # 해당 index의 유사도 리스트 sort in descending order
    score_series = pd.Series(new_sim[idx[0]]).sort_values(ascending = False)

    # 유사도 Top 10의 index 추출
    top_10_indexes = list(score_series.iloc[1:11].index)

    # 유사도 1인 항목이 하나 더 있어서 자기 자신이 포함되는 경우에는 자신을 뺀 Top 10의 index 재추출
    if top_10_indexes[0] == idx[0]:
        top_10_indexes = list(score_series.iloc[1:12].index)
        top_10_indexes.remove(idx[0])

    # 고유 id를 담기 위한 empty list 생성
    top_10_id = []

    # id list
    for i in top_10_indexes:
        id = data.loc[i]['id']
        top_10_id.append(id)

    return top_10_id

# test
print(general_recommendation(10))
```



[323, 90, 383, 92, 192, 557, 136, 349, 397, 360]



## 각자 구현한 코드

### 1) 서영 - Euclidean Distance + Cosine Similarity

```
[ ] # wine word list 만들기
wine_list = [] # empty list

# wine index list 추출
wine_idx = data.index[data['class'] == "wine"].tolist()

# wine: category, percent, origin, producer, wine_grape, flavour
for i in wine_idx:
    temp = ""
    temp = temp + data.loc[i]['category'] + " " + data.loc[i]['origin'] + " " + data.loc[i]['producer'].replace(" ", "") + " " + data.loc[i]['wine_grape'].replace(" ", "") + " "
    for flavour in data.loc[i]['flavour']:
        temp = temp + flavour
    wine_list.append(temp)

# wine 코사인 유사도 구하기
# instantiating and generating the count matrix
count = CountVectorizer()
wine_matrix = count.fit_transform(wine_list)

# generating the cosine similarity matrix
wine_sim = cosine_similarity(wine_matrix, wine_matrix)

# wine 도수 유사도 구하기
wine_pair = []
```

텍스트의 코사인 유사도를 구하기 위해  
변수를 텍스트화



# 각자 구현한 코드

## 1) 서영 - Euclidean Distance + Cosine Similarity

```
# y축을 임의로 0을 부여한 거리 순서쌍 생성
for i in wine_idx:
    temp = []
    temp.append(data.loc[i]['percent'])
    temp.append(0)
    wine_pair.append(temp)

# get a distance matrix
wine_df = pd.DataFrame(wine_pair, columns=['x', 'y'])
wine_matrix = distance_matrix(wine_df.values, wine_df.values)

# 정규화
min_max_scaler = MinMaxScaler()
wine_regularised = min_max_scaler.fit_transform(wine_matrix)

# 1에서 빼줘서 더 가까운 것이 우선순위를 갖도록 변경하기
wine_one_matrix = np.ones((len(wine_idx), len(wine_idx)))

wine_final_dist = wine_one_matrix - wine_regularised

# 도수 유사도, 타 정보 유사도 각각 0.5씩 weight 부여 후 새로운 matrix 생성
wine_new_sim = 0.5 * wine_sim + 0.5 * wine_final_dist

# 확인
print(wine_new_sim)
```

[[1. 0.71660997 0.65957047 ... 0.65957047 0.55027799 0.63608276] [0.70966553 1. 0.70204326 ... 0.62489159 0.43589744 0.60416667] [0.65773224 0.70367725 1. ... 0.64285714 0.38484398 0.62305335] ... [0.65773224 0.62652557 0.64285714 ... 1. 0.38484398 0.54590167] [0.57912415 0.5 0.43009285 ... 0.43009285 1. 0.45833333] [0.63608276 0.61111111 0.62489159 ... 0.54773991 0.42948718 1. ]]

```
[ ] # 해당 id의 술이 wine인지 체크
def is_wine(input_id):
    temp_idx = data.index[data['id'] == input_id].tolist() # Int64Index 형식이라 list로 바꾸어줌
    result = data.loc[temp_idx[0]]['class'] == "wine"
    return result

# 와인의 고유 id를 넣으면 해당 와인과 비슷한 Top 10 와인의 id를 return
def wine_recommendation(input_id, wine_new_sim = wine_new_sim):

    # 고유 id로 index 찾기
    idx = data.index[data['id'] == input_id].tolist() # Int64Index 형식이라 list로 바꾸어줌

    # wine_idx list 내에서 몇번째 와인인지 구하기
    w_idx = wine_idx.index(idx[0])

    # 해당 index의 유사도 리스트 sort in descending order
    score_series = pd.Series(wine_new_sim[w_idx]).sort_values(ascending = False)

    # 유사도 Top 10의 index 추출
    wine_top_10_indexes = list(score_series.iloc[1:11].index)

    # 유사도 1인 항목이 하나 더 있어서 자기 자신이 포함되는 경우에는 자신을 뺀 Top 10의 index 재추출
    if wine_top_10_indexes[0] == w_idx:
        wine_top_10_indexes = list(score_series.iloc[1:12].index)
        wine_top_10_indexes.remove(w_idx)

    # 고유 id를 담기 위한 empty list 생성
    wine_top_10_id = []

    # id list
    for i in wine_top_10_indexes:
        index = wine_idx[i] # wine list에서 몇번째인지가 아니라 전체 술 list에서 몇번째인지 구함
        id = data.loc[index]['id']
        wine_top_10_id.append(id)

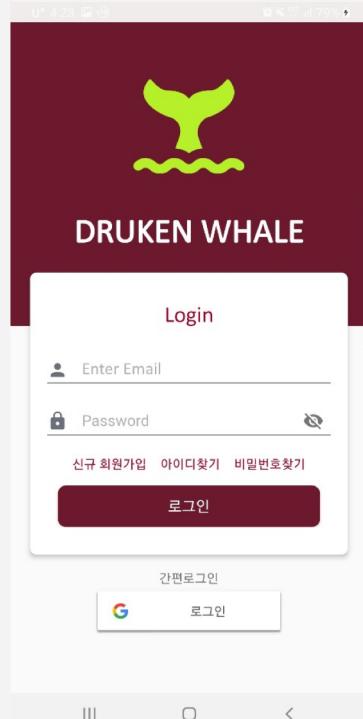
    return wine_top_10_id
```



주류 추천 어플리케이션 술고래

04 앱 구성 및 시연

## 앱 구성



1. 로그인
2. 회원가입
3. 아이디/비밀번호 찾기
4. 구글 간편 로그인



## 앱 구성



1. 술 이름 및 주종 대분류, 중분류를 통한 검색 기능
2. 내가 쓴 평점에 대한 기능 (개발예정)



## 주류 추천 어플리케이션 술고래

### 04 앱 구성 및 시연

## 앱 구성



1. 주종에 따른 커뮤니티 구성
2. 전체/기타 게시판
3. 글쓰기



## 주류 추천 어플리케이션 술고래

04 앱 구성 및 시연

### 앱 구성



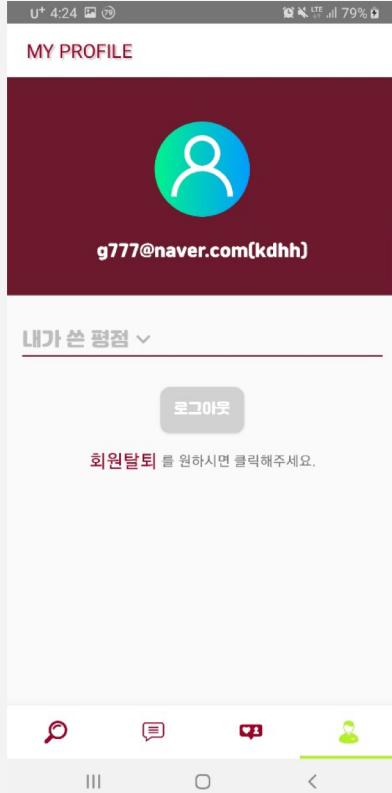
주종 및 취향에 따른 사용자의 즐겨찾기 기능



주류 추천 어플리케이션 술고래

04 앱 구성 및 시연

## 앱 구성



1. 프로필 이미지 설정
2. 로그아웃
3. 회원탈퇴



## 주류 추천 어플리케이션 술고래

04 앱 구성 및 시연

### 앱 구성



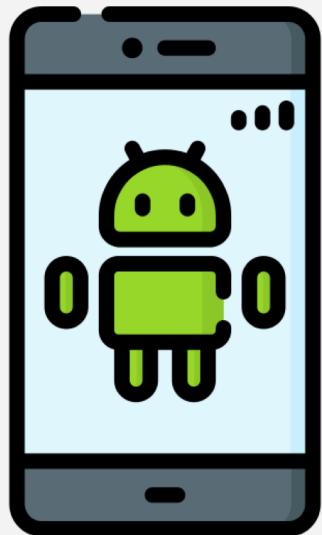
1. 술 주종 및 이름에 따른 도수, 가격 등 상세정보를 보여주는 기능
2. 해당 주종에 맞는 추천 주종을 보여주는 기능
3. 평점 기능(개발예정)



주류 추천 어플리케이션 술고래

04 앱 구성 및 시연

## 앱 시연





## 향후 서비스 방향성

- 주류 데이터베이스에 대한 고도화가 필요함
- 사용자 정보가 누적되면, 이를 분석해 개인의 특성에 맞는 술을 추천
- 평점 분석 기능 추가



THANK YOU  
FOR LISTENING!