Dacon Competition Trends with keyword & clustering

작성자: 김효정, 이우진

CONTENTS

01

프로젝트 개요

01 프로젝트 개요

DACON AI 경진대회 데이터 기반 분석

2018-2023년의 경진대회 트랜드 분석 및 발전방향 제안

사용 데이터(Dacon 제공)

- **Competition**: 데이콘에서 진행된 공개 AI 경진대회 관련 정보 (대회ID, 대회기간, 대회키워드, 상금, 참가조건 등)
- **Submission**: 특정 대회에서 발생한 제출 수 관련 정보
- **Participate**: 특정 대회에서 발생한 참가자 수 관련 정보
- **Talk**: 특정 대회에서 발생한 토크 관련 정보
- **Codesharing**: 특정 대회에서 발생한 코드 공유 관련 정보

01 프로젝트 개요

데이터셋 → 시각화 → Clustering → 결과 분석 및 결론

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
plt.rc('font', family='Malgun Gothic')

import warnings
warnings.filterwarnings('ignore')
```

<사용한 python 패키지>

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn import set_config
set_config(transform_output='pandas') # 파이프라인 통과한 출력물 형식을 데이터프레임으로 고정

import scipy.cluster.hierarchy as sch
from sklearn.decomposition import PCA
import plotly.express as px
```

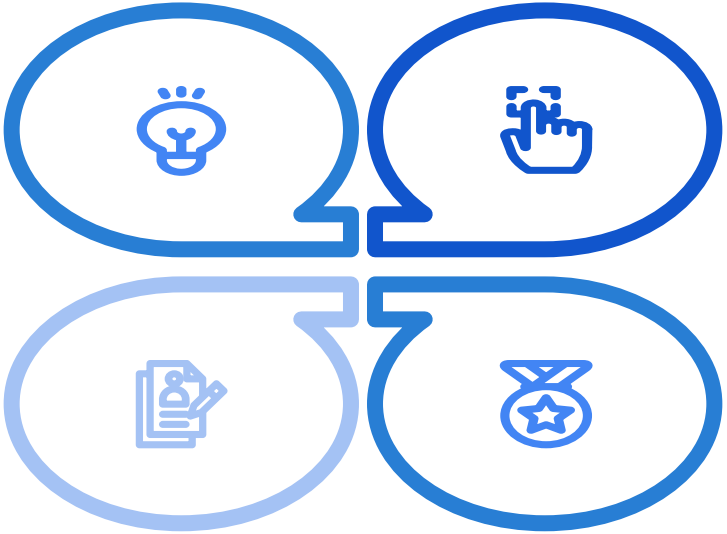
02

시각화 분석

02

시각화 분석

키워드
키워드에서 제일 높은 빈도수를 가진 조합 5개를 가지고 분석



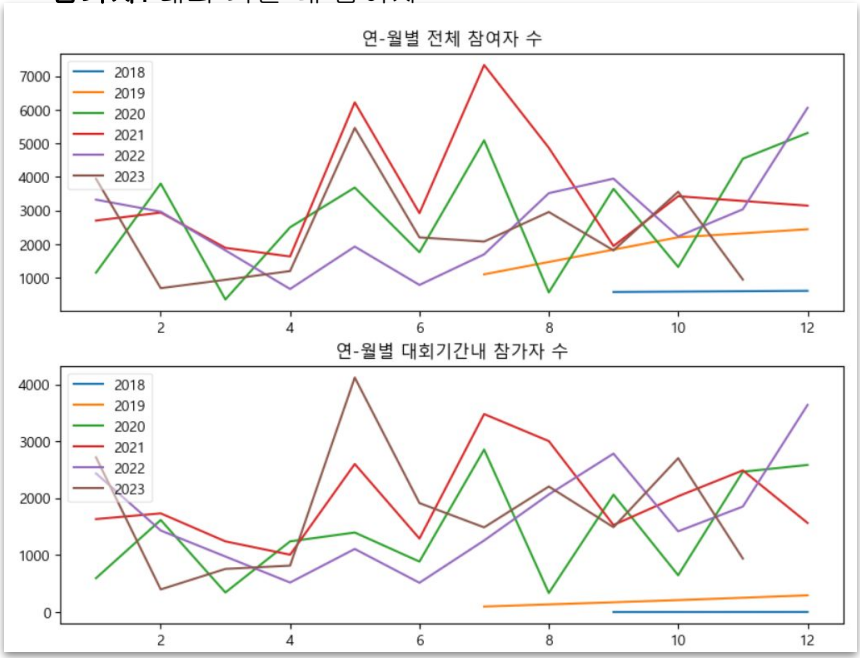
분야
임의로 4가지 분야(사회 ,과학, 환경, 기타)로 나누어 분석

채용
‘채용’ 유무에 따라 분석

상금
‘상금’ 유무에 따라 분석

02 시각화 분석 - 참여자 & 참가자 수

참여자: 대회 시작부터 대회 종류 이후 현재까지 총 참여자
참가자: 대회 기간 내 참여자



<연월별 참여자/참가자 수 그래프 분석 결과>

1) 데이콘 초반 단계여서 대회 수가 적어 **2018**년도와 **2019**년도는 연도 대회는 참가율이 저조 한 것을 확인

2) **2021**년, **2023**년에 참여자수, 참가자수 둘 다 높은 것으로 확인

3) 대체적으로 **5**월, **7**월과 **12**월에 참가율이 좋은 것을 보아 학생들의 참여가 좋을 것이라 추측

4) 위 추측을 기반으로 대회에 '채용'관련 키워드의 유무가 영향이 있을 것이라 예상

02 시각화 분석 - 키워드

<상위 다섯개 키워드로 대회 당 키워드 수와 순서 확인>

```
0      알고리즘,정형,회귀,금융,RMSE
1      알고리즘,정형,회귀,아파트,RMSE
2     알고리즘,정형,회귀,스포츠,WRMSE
3     알고리즘,정형,회귀,매출,회귀,MAE
4      알고리즘,정형,회귀,공공,SMAPE
Name: 키워드, dtype: object
```

<대체적으로 키워드가 순서에 따라 특징을 보이는 것으로 확인>

첫번째 키워드는 주로 '알고리즘', '분석시각화'로 분석방식의 큰 틀이 주어짐
두번째 키워드는 '정형', 'NLP', '비전' 등 데이터 종류를 암시
세번째 키워드는 '회귀', '분류', '분석' 등으로 분석방법을 제시
네번째 키워드는 데이터의 특징을 주로 알려주지만 각각을 구분하기에는 종류가 너무 다양함
다섯번째 키워드는 대회 평가 방식

대회마다 대부분 5개의 키워드를 가지지만 마지막 키워드는 주로 채점방식이기 때문에 제외하고 대회당 4개의 키워드를 분석

```
kw_arr shape: (145, 4)
1 번째 키워드
Counter({'알고리즘': 119, '분석시각화': 16, '시계열': 2, '채용': 2, '아이디어': 2, '운동': 1, 'SW중심대학': 1, '분석아이디어': 1, '제주도': 1})
2 번째 키워드
Counter({'정형': 76, 'CV': 14, 'NLP': 10, '비전': 10, '이미지': 6, '언어': 5, '텍스트': 4, 'Audio': 3, '음향': 2, '채용': 2, '알고리즘': 2, 'KeypointDetection': 1, 'SMAPE': 1,
'정형&비정형': 1, '본선': 1, '관광': 1, '유전체': 1, '동영상': 1, 'ChatGPT': 1, '금융': 1, '멀티모달': 1, '분자구조': 1})
3 번째 키워드
Counter({'회귀': 43, '분류': 42, '분석': 14, '객체탐지': 4, '언어': 4, '정형': 4, '생성요약': 3, '시계열': 3, '이상탐지': 2, '비전': 2, '생육': 2, '교통': 2, '객체검출': 1,
'강화학습': 1, '영상분할': 1, 'Semanticsegmentation': 1, '한국에너지공단': 1, '극지연구소': 1, '이미지변환': 1, '시각화': 1, 'ImageSuper-Resolution': 1, 'OCR': 1, '산업': 1,
'건설기계': 1, '비지도학습': 1, '광학문자인식': 1, '동영상': 1, '프롬프트엔지니어링': 1, 'AI': 1, '감정인식': 1, '이미지분할': 1, '추천시스템': 1})
4 번째 키워드
Counter({'금융': 15, '제어': 9, '교통': 5, '산업': 5, '자연어': 5, '에너지': 4, '회귀': 4, 'Accuracy': 4, '분류': 4, '언어': 3, '행태심리': 3, '환경': 3, '공공': 2, '과학': 2,
'기상': 2, '물성': 2, '농작물': 2, '수요예측': 2, '검출': 2, '생육': 2, '추천': 2, '탐지': 2, 'Macrof1score': 2, '비전': 2, '아파트': 1, '스포츠': 1, '매출': 1, '우주': 1, '게
```

02 시각화 분석 - 키워드

<5개의 조합>

1. 알고리즘,정형,회귀
2. 알고리즘,비전,분류
3. 알고리즘,언어,분류
4. 알고리즘,정형,분류
5. 분석시각화,정형,분석

```
알고리즘,정형,회귀 43
2018: 2 / 2019: 5 / 2020: 11 / 2021: 13 / 2022: 8 / 2023: 4 /
알고리즘,비전,분류 19
2018: 0 / 2019: 0 / 2020: 6 / 2021: 5 / 2022: 4 / 2023: 4 /
알고리즘,언어,분류 16
2018: 0 / 2019: 0 / 2020: 4 / 2021: 4 / 2022: 5 / 2023: 3 /
알고리즘,정형,분류 12
2018: 0 / 2019: 0 / 2020: 3 / 2021: 5 / 2022: 3 / 2023: 1 /
분석시각화,정형,분석 16
2018: 0 / 2019: 0 / 2020: 6 / 2021: 6 / 2022: 4 / 2023: 0 /
기타 43
2018: 0 / 2019: 0 / 2020: 4 / 2021: 10 / 2022: 11 / 2023: 18 /
```

<키워드별 데이터 분석 결과>

1) 알고리즘,정형,회귀 키워드는 대회 수가 점점 줄어들은 것을 확인하였으나 그 외 4가지 키워드의 수는 변화가 크지 않음을 확인

2) 최근 이미지캡셔닝, 이미지분할, 시계열 등 이전에 존재하지 않았던 키워드를 가지고 있는 대회가 생겨남을 확인

02 시각화 분석 - 분야

```
cat_dic = {'사회': ['금융', '교통', '산업', '공공', '수요예측', '코로나', '선거',
                  '제조', '유동인구', '상권', '전력수요량', '사회', '부동산', '운송량',
                  '의료', '서비스', '해외주식분석', '관광', '건설기계'],
          '과학': ['제어', '에너지', '과학', '물성', '우주', '바이오', '영상분할',
                  '이미지세분화', '시스템', '전력', '자율주행', '3D', '초전도체',
                  '도메인적응', '탐지', '유전체', '광학문자인식', 'AI', '분자구조',
                  '생성요약', 'OCR', '검출', 'ChatGPT', '프롬프트 엔지니어링', 'MLOps',
                  '객체탐지', '멀티모달', '컴퓨터비전', '영상', 'SEM', 'ImageSuper-Resolution',
                  'Audio', '이미지캡셔닝', '유사성'],
          '환경': ['환경', '기상', '농작물', '생육', '강우예측', '북극', '농산물', '기후기술'],
          '기타': ['행태심리', '아파트', '스포츠', '매출', '게임', '심리', '헬스', '트렌드',
                  '포트폴리오구성', '조선해양', '데이크루', '채용', '이상탐지', '음향',
                  '운동', '항공', '플랫폼']}
```

```
사회 43
2018: 1 / 2019: 3 / 2020: 15 / 2021: 14 / 2022: 9 / 2023: 1 /
기타 42
2018: 1 / 2019: 2 / 2020: 4 / 2021: 7 / 2022: 12 / 2023: 16 /
과학 49
2018: 0 / 2019: 0 / 2020: 12 / 2021: 15 / 2022: 9 / 2023: 13 /
환경 15
2018: 0 / 2019: 0 / 2020: 3 / 2021: 7 / 2022: 5 / 2023: 0 /
```

<연별 분야별 참여자 수 분석 결과>

1) 대회나 데이터의 특징을 설명하는 키워드에서 임의로 **4**가지 분야로 나누어 키워드를 분리(사회, 과학, 환경, 기타)

2) 임의로 나눈 분야에 따라 '과학', '기타' 분야의 참가자들이 가장 많은 것을 확인

3) '사회', '환경'은 **2021-2023**까지는 대회 수가 적지 않았으나 **2023**년에는 비교적 저조한 것으로 확인

4) 대회의 트렌드가 '기타'와 '과학으로 바뀐 것으로 추정

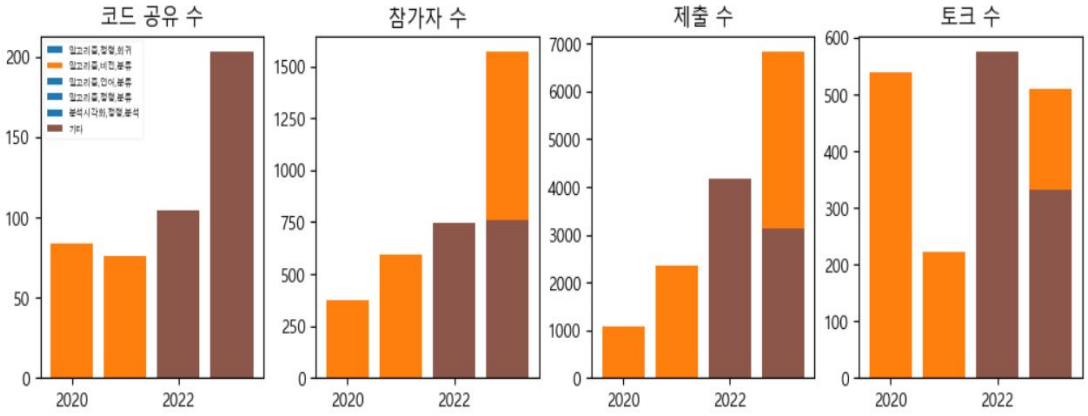
02 시각화 분석 - 채용

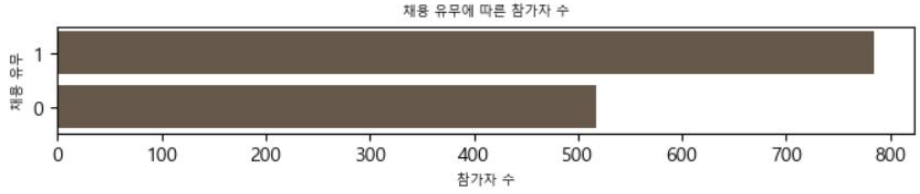
```
- 키워드 칼럼의 채용
Int64Index([46, 121, 125, 131, 146], dtype='int64')
- 상금 상세정보 칼럼의 채용
Int64Index([27, 114, 121, 125, 126, 129, 130, 131, 146], dtype='int64')
```

		ID
keyword	year	
기타	2022	1
	2023	6
	2020	1
알고리즘,비전,분류	2021	1
	2023	1

- '키워드'칼럼에는 '채용'이 4개 존재하지만 '상금 상세정보'칼럼에 '채용'이라는 단어가 있는지 검색하면 9개의 데이터 존재
  - 총 10개의 채용 키워드 확인
- '채용'여부에 따른 변화 추이를 보기 위해 앞선 키워드 분석 그래프에 '채용' 키워드 데이터를 생성하고 확인

02 시각화 분석 - 채용





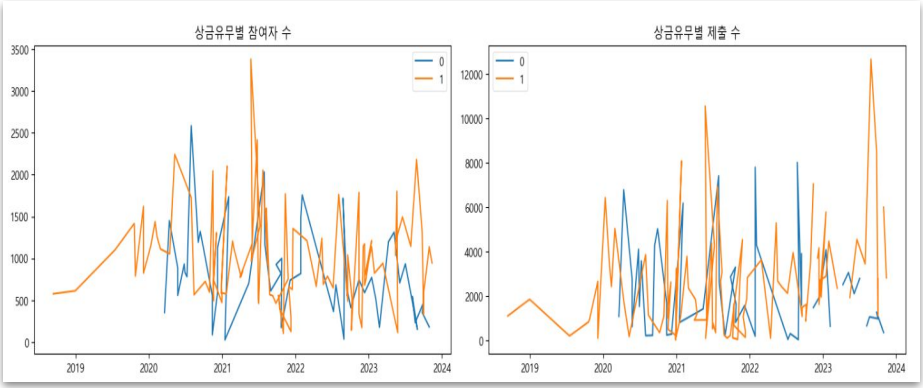
<채용 키워드 데이터 분석 결과>

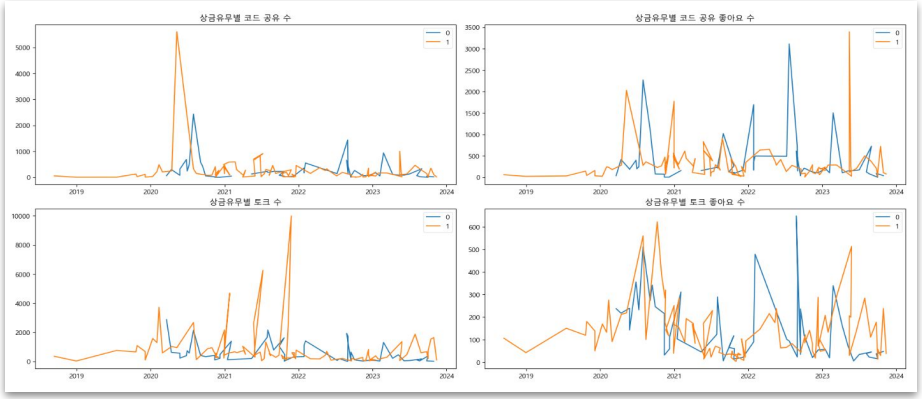
1) 앞선 키워드 구분에서 '알고리즘,비전, 분류'와 '기타'에 채용 키워드 존재 확인

2) 채용이 있는 대회의 참가자 수가 높은 것을 확인(채용유: **805**명, 채용무: **519** (평균치계산))

02 시각화 분석 - 상금

```
# 상금 유무 컬럼을 추가
temp_competition['상금'] = df['competition']['상금 정보'].apply(lambda x: 0 if x in ['인증서', '000만원', np.nan] else 1)
```





<상금 데이터셋 설명>

- 상금정보 칼럼에 '인증서,'0원','null'을 이외에는 상금으로 취급
- 상금 유무에 따라 데이터 프레임 'competition', 'submission', 'codesharing', 'talk' 비교

<상금유무로 나눈 데이터 분석 결과>

1) 상금 유무에 따른 대회의 비율은 **1:2**로 확인

2) 상금이 있는 대회가 없는 대회보다 참여자 수, 제출 수, 코드 공유 수 등 모든 경우에 값이 높음

3) 위 추측을 기반으로 대회에 '상금' 관련 키워드의 유무가 영향이 있을 것이라 예상

03

클러스터링 분석

03 클러스터링 분석 - Pipeline(전처리)

Class CreateX(): 분석에 앞서 여러 파일에 흩어져 있는 데이터를 하나로 합침
ChooseCols()에서 가공 될 초기 데이터프레임 제공

```
# Pipeline에 넣을 클래스 1
# 분석에 앞서 여러 파일에 흩어져 있는 데이터를 하나로 합쳐줌
# ChooseCols()에서 가공될 초기 데이터프레임 제공
class CreateX(BaseEstimator, TransformerMixin):
    def __init__(self, df):
        self.df = df

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):
        compe_cols = ['ID', '참여자 수', 'year', 'month']
        X = self.df['competition'][compe_cols].copy()
        X.loc[:,'keyword'] = self.col_keyword()
        X.loc[:,'분야'] = self.col_분야()
        X.loc[:,'채용'] = 0
        X.loc[self.df['competition'][df['competition']['상금 상세정보'].str.contains('채용')].index, '채용'] = 1
        X.loc[:,'상금'] = self.df['competition']['상금 정보'].apply(lambda x: 0 if x in ['인증서', '000만원', np.nan] else 1)

        temp_dic = {'codesharing':'코드 공유 수', 'participate':'참가자 수',
                    'submission':'제출 수', 'talk':'토크 수'}
        for k,v in temp_dic.items():
            X = pd.merge(X, self.df[k].groupby('ID')[v].sum(), on='ID', how='left')
        return X.fillna(0)
```

```
    def col_keyword(self): # 'keyword' 칼럼 생성
        temp = self.df['competition'].copy()
        change_dic = {'언어': ['텍스트', 'NLP', '자연어'],
                      '비전': ['CV', '이미지', '객체검출', '객체탐지']}
        for k,vs in change_dic.items():
            for v in vs:
                temp['키워드'] = temp['키워드'].apply(lambda x: x.replace(v,k))
        combi = {'알고리즘,정형,회귀':45, '알고리즘,정형,분류':12,
                 '알고리즘,언어,분류':16, '알고리즘,비전,분류':19,
                 '분석시각화,정형,분석':16}
        temp['keyword'] = None
        for comb, n in combi.items():
            idx = similarity(temp, '키워드', comb, n, 'ascending').index
            temp.loc[idx, 'keyword'] = comb
        temp['keyword'] = temp['keyword'].fillna('기타')
        return temp['keyword']
```

```
예측', '코로나', '선거',
사회', '부동산', '운송량',
건설기계'],
'바이오', '영상분할',
행', '3D', '초전도체',
인식', 'AI', '분자구조',
프롬프트 엔지니어링', 'MLOps',
상', 'SEM', 'ImageSuper-Resolution',

측', '북극', '농산물', '기후기술'],
'게임', '심리', '헬스', '트렌드',
, '채용', '이상탐지', '음향',
'운동', '항공', '플랫폼']}
        temp['분야'] = None
        for k,vs in cat_dic.items():
            for v in vs:
                idx = self.df['competition'][self.df['competition']['키워드'].str.contains(v)].index
                temp.loc[idx, '분야'] = k
        temp['분야'] = temp['분야'].fillna('기타')
        return temp['분야']

    def get_feature_names_out(self, feature_names_in):
        return feature_names_in
```

03 클러스터링 분석 - Pipeline(전처리)

Class ChooseCols(): 분석에 사용할 칼럼 선택

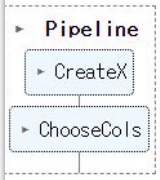
```
# Pipeline에 넣을 클래스 2
class ChooseCols(BaseEstimator, TransformerMixin):
    def __init__(self, kw=0, sptc=[False, False, False, False], drop_cols=[]):
        self.kw = kw
        # kw: 0이면 분석에 키워드를 사용하지 않고, 1이면 'keyword'를 통해 분석방식마다 구분
        # 2면 '분야'를 통해 데이터의 특징을 구분
        self.sptc = sptc #submission, participate, talk, codeshareing으로 False인 경우 해당 칼럼을 drop
        self.drop_cols = drop_cols # 이 외 추가로 지우고 싶은 칼럼 drop

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):
        self.drop_cols.append('ID')
        for check, col in zip(self.sptc, ['제출 수','참가자 수','토크 수','코드 공유 수']):
            if not check:
                self.drop_cols.append(col)
        if self.kw==0:
            self.drop_cols += ['분야','keyword']
        elif self.kw==1:
            X = pd.get_dummies(X, columns=['keyword'], drop_first=True)# pd.dummies를 이용해 원핫인코딩 진행
            self.drop_cols.append('분야')
        elif self.kw==2:
            X = pd.get_dummies(X, columns=['분야'], drop_first=True) # pd.dummies를 이용해 원핫인코딩 진행
            self.drop_cols.append('keyword')
        return X.drop(columns=self.drop_cols)

    def get_feature_names_out(self, feature_names_in):
        return feature_names_in
```

<ChooseCols: kw, sptc, drop_cols>

- Kw: kw=0이면 분석에 키워드를 사용하지 않고, 1이면 ‘keyword’를 통해 분석, 2이면 ‘분야’에 따라 분석
- sptc: 순서대로 submission, participate, talk, codesharing의 사용여부를 뜻하며 False일 경우 해당 칼럼을 drop
- drop_cols: 추가로 삭제할 칼럼 지정

03 클러스터링 분석 - Pipeline(전처리)

```
# 파이프라인
# CreateX(), ChooseCols() 두 개의 class를 이용해 preprocessing이라는 파이프라인 생성
preprocessing = Pipeline([
    ('CreateX', CreateX(df)),
    ('ChooseCols', ChooseCols(0, [True, True, True, False], [])),
            ])
display(preprocessing)
X = preprocessing.transform(df)
X.head()
```

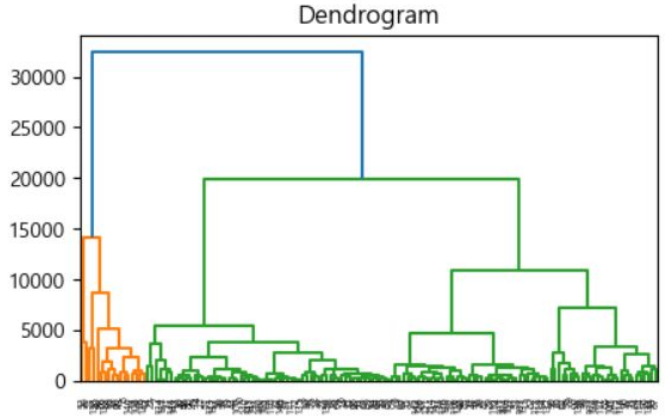


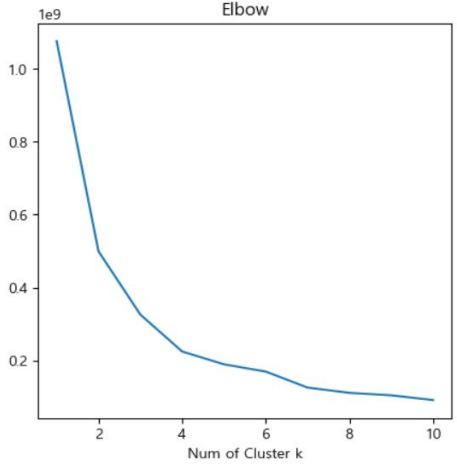
- CreateX(), ChooseCols() class를 이용해 preprocessing이라는 Pipeline을 생성하고 이를 데이터 전처리에 사용
- ChooseCols에 있는 옵션을 지정해 원하는 전처리 가능

03 클러스터링 분석 - 군집 수 결정

k-means 사용에 앞서 최적의 군집개수 k를 찾는 방법

- Dendrogram: 계층적 군집분석시 단계별로 계층을 따라가며 군집을 합쳐가는 과정을 보여주는 그래프, 적정선에 가로선을 그어 군집 개수를 결정
- Elbow: 군집 간의 거리의 합을 나타내는 inertia가 급격히 떨어지는 구간이 생기는데 그 지점을 k값(군집 개수)로 사용 inertia_속성으로 확인
- Silhouette: 군집타당성지표인 실루엣 점수 이용, 1에 가까울 수록 적절한 군집화가 되었다고 판단





03 클러스터링 분석 - 군집 수 결정

Dendrogram

```
def draw_Dendrogram(X): # Dendrogram 그리기
    fig = plt.figure(figsize=(5,3))
    cluster_visualising=sch.dendrogram(sch.linkage(X.values, method='ward'))
    plt.title('Dendrogram')
    plt.show()

draw_Dendrogram(X)
```

Elbow

```
def draw_Elbow(X, title=None): # Elcow 그래프로 최적의 k 찾기
    fig = plt.figure(figsize=(5,5))
    min_dist = []
    for k in range(1,11):
        clust = KMeans(n_clusters=k, n_init='auto')
        clust.fit(X)
        min_dist.append(clust.inertia_)
    plt.plot(range(1,11), min_dist)
    plt.title(title) if title is not None else plt.title('Elbow')
    plt.xlabel('Num of Cluster k')
    plt.show()

draw_Elbow(X)
```

Silhouette Score

```
def useSilhouette(X): # Silhouette score를 이용해 최적의 k 구하기
    best_k, best_score = -1, -1
    for k in range(2,10):
        clust = KMeans(n_clusters=k, n_init='auto')
        clust.fit(X)
        labels = clust.predict(X)
        score = silhouette_score(X, labels)
        if score>best_score:
            best_k, best_score = k, score
    return best_k

useSilhouette(X)
```

```
3
```

03 클러스터링 분석 - KMeans Clustering

```
# Elbow로 구한 최적 k는 4, Silhouette score로 구한 최적 k는 2
k_elbow, k_silhouette = 4, 2
cluster_elbow = KMeans(n_clusters=k_elbow, n_init='auto')
cluster_silhouette = KMeans(n_clusters=k_silhouette, n_init='auto')
labels_elbow = cluster_elbow.fit_predict(X)
labels_silhouette = cluster_silhouette.fit_predict(X)
print('labels_elbow:', np.unique(labels_elbow, return_counts=True))
print('labels_silhouette:', np.unique(labels_silhouette, return_counts=True))
```

```
labels_elbow: (array([0, 1, 2, 3]), array([84, 15, 47,  3], dtype=int64))
labels_silhouette: (array([0, 1]), array([ 40, 109], dtype=int64))
```

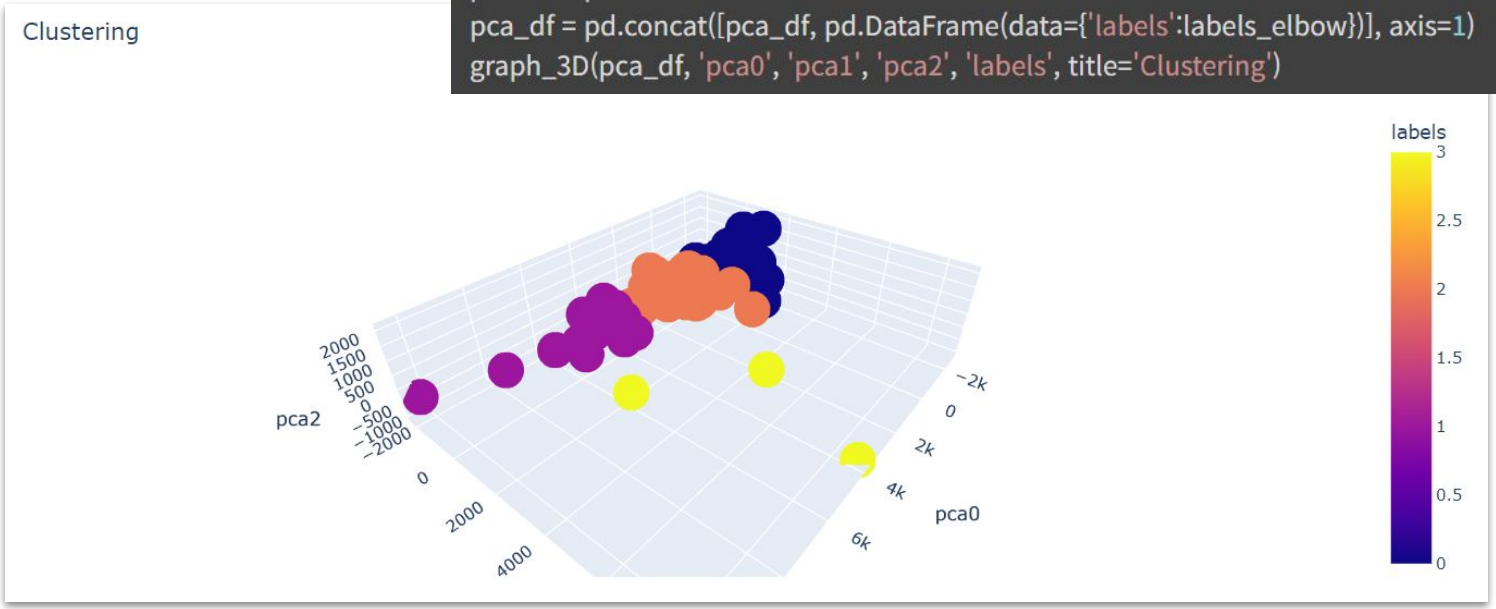
- **K-Means Clustering 결과**
  - Elbow 그래프를 통해 구한 최적 n_clusters(k)는 4
  - Silhouette Score를 이용해 구한 최적 n_clusters(k)는 2
  - 각각의 n_clusters에 따라 군집화한 결과 4개로 구분된 군집들에는 각각 84, 15, 47, 3개의 데이터가 들어가고 2개로 나누어진 구집에는 40, 109개의 데이터가 들어감

03 클러스터링 분석 - KMeans Clustering

```
# 분석한 데이터의 칼럼 수가 많기 때문에 시각화 편의를 위해 pca로 차원축소(3차원)
def graph_3D(plot_df, xn, yn, zn, c, title): # 반응형 3d 그래프 그리기
    fig = px.scatter_3d(plot_df, x=xn, y=yn, z=zn, color=c, title=title)
    fig.show()
```

- K-Means Clustering 결과 3D-Scatter Plot
  3D 그래프 출력을 위해 PCA 선행

```
pca = PCA(n_components=3)
pca_df = pca.fit_transform(X) # X의 전체 칼럼들의 특징을 반영한 3개의 주성분으로 구성된 데이터프레임 생성
pca_df = pd.concat([pca_df, pd.DataFrame(data={'labels':labels_elbow})], axis=1)
graph_3D(pca_df, 'pca0', 'pca1', 'pca2', 'labels', title='Clustering')
```



03 클러스터링 분석

<군집들의 데이터 분포 확인>

```
1 : useElbow [114, 32, 3] / useSilhouette [116, 33]
3 : useElbow [84, 49, 16] / useSilhouette [109, 40]
18 : useElbow [84, 47, 16, 2] / useSilhouette [116, 33]
19 : useElbow [84, 49, 16] / useSilhouette [116, 33]
33 : useElbow [84, 49, 16] / useSilhouette [116, 33]
35 : useElbow [84, 49, 16] / useSilhouette [109, 40]
```

군집분석 실행

- 만들어둔 Preprocessing Pipeline의 ChooseCols() 옵션을 통해 전처리를 다르게 하여 각 경우마다 최적의 k값을 구하고(Elbow, Silhouette 이용) Kmeans 군집화 실행
- kw와 sptc를 바꿔가며 군집분석을 실행해 3D 그래프를 확인해 본 결과 1,3,18,19,33,35 번째의 시도에서 그룹들이 섞이지 않고 잘 나뉘어져 있는 것을 확인
- 각 군집들의 데이터 분포를 출력해 본 결과 1, 18의 경우 가지고 있는 데이터가 5개 이하인 작은 군집이 있기에 최종적으로 군집화가 잘 된 경우는 3, 19, 33, 35
- 3, 19, 33, 35번째 시도에서 실행했던 전처리 조건을 확인한 결과로 군집분석 결론 도출

<전처리 조건>

```
3 : [0, True, True, False, True]
19 : [1, True, True, False, True]
33 : [2, True, True, True, True]
35 : [2, True, True, False, True]
```

군집분석 결과

- 키워드가 포함되어 있는 결과보다 분야가 포함되어 있는 결과가 더 좋은 군집을 이룸
- 제출 수와 참가자 수, 그리고 코드공유 수는 항상 있는 것이 유리
- talk 수는 그다지 군집형성에 큰 영향을 미치지 않음

04

결론 및 제안사항

04 결론 및 제안사항

결론

'회귀' 키워드를 가진 대회는 꾸준히 있었지만 '비전', '언어', '분석' 등의 키워드를 가진 대회는 2020년 이후 부터 등장하였습니다. 최근에는 분석시각화나 아이디어 경진대회 등이 포함된 '기타' 카테고리의 대회 수가 증가하였습니다.

결론

데이터 분야(사회, 과학, 환경, 기타)로 분석 결과 연간 각 분야별 참여자 수 증감추이는 비슷하지만 2023년에 한하여 과학과 기타(‘포트폴리오’, ‘채용’, ‘트랜드’, ‘심리’ 등) 분야의 대회가 증가하였습니다.

제안 사항

기존 대회에는 연습으로 참여가 가능하므로, 기존에 없던 종류의 대회를 꾸준히 증가시키는 것을 제안드립니다.

04 결론 및 제안사항

결론

시기별 참가자 수 분석결과 5,7,12월에 참가자 수가 많은것을 보아 학생들이 중간고사 이후 또는 방학 시즌에 참가를 많이 하는것을 추측됩니다.

제안 사항

위 결론을 토대로 학생 참가자들이 많은 것으로 추측되어 추후 대회는 학생들이 관심을 가질 수 있는 특징(상금,채용), 분야를 가진 대회를 늘리는 것을 제안드립니다.

Thank you