# 머신러닝 기술을 이용한 개인의 특성에 따른 패션 추천

이재호(B811141, 컴퓨터공학전공) [1] 지도교수: 송하윤(컴퓨터공학전공) [2]

홍익대학교

## Introduction

본 연구의 목적은 머신러닝 기술을 활용하여, 개인의 특성(예: 성별, 나이, 직업, 선호 스타일 등)에 맞게 패션을 추천해 주는 인공지능 모델을 구축하는 것이다. 본 연구에서는 머신러닝 기술 중 Logistic Regression, AdaBoost Classifier, Gradient Boosting Classifier을 사용하였으며, 참가자들의 설문 조사 dataset을 가지고 그 결과를 확인하였다. 이때 설문 조사 dataset은 참가자들에게 패션 이미지를 보여주고 해당 이미지에 대한 평가와 참가자들의 정보를 담고 있는 dataset이다. 끝으로, 전체적인 과정은 '1) 데이터 전처리, 2) 모델링, 3) 결과'로 정리하였다.

## 1. 데이터 전처리

raw dataset에서 이미지 dataset 및 참가자 dataset 추출 및 가공하는 것을 목적으로 하였다.

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | E_id | imgName | era | style | gend | Q1 | Q2 | Q3 | Q41 | Q41 | Q41 | Q41 | Q420 | Q420 |
| 2 | 1 | W_15268_50_ivy_M.jpg | 1950 | ivy | M | 4 | 1 | 1 | 3 | 2 | 2 | 2 | 0 | 2 |
| 3 | 2 | W_16543_50_ivy_M.jpg | 1950 | ivy | M | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 0 | 0 |
| 4 | 3 | W_17697_50_ivy_M.jpg | 1950 | ivy | M | 3 | 1 | 1 | 2 | 2 | 2 | 2 | 0 | 2 |
| 5 | 4 | W_00485_60_mods_M.jpg | 1960 | mods | M | 4 | 1 | 1 | 2 | 2 | 2 | 2 | 1 | 0 |
| 6 | 5 | W_06723_60_mods_M.jpg | 1960 | mods | M | 3 | 1 | 5 | 3 | 1 | 1 | 1 | 0 | 0 |
| 7 | 6 | W_15212_60_mods_M.jpg | 1960 | mods | M | 2 | 1 | 2 | 1 | 1 | 1 | 1 | 0 | 0 |
| 8 | 7 | W_01612_70_hippie_M.jpg | 1970 | hippie | M | 1 | 1 | 5 | 1 | 1 | 1 | 1 | 0 | 0 |
| 9 | 8 | W_03007_70_hippie_M.jpg | 1970 | hippie | M | 4 | 1 | 5 | 2 | 2 | 2 | 2 | 0 | 2 |
| 10 | 9 | W_06563_70_hippie_M.jpg | 1970 | hippie | M | 1 | 2 | 4 | 3 | 2 | 1 | 2 | 0 | 0 |
| 11 | 10 | W_15371_80_bold_M.jpg | 1980 | bold | M | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 12 | 11 | W_16068_80_bold_M.jpg | 1980 | bold | M | 3 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 0 |
| 13 | 12 | W_17957_80_bold_M.jpg | 1980 | bold | M | 3 | 1 | 2 | 2 | 1 | 1 | 1 | 0 | 2 |
| 14 | 13 | W_02785_90_hiphop_M.jpg | 1990 | hiphop | M | 1 | 2 | 5 | 2 | 2 | 1 | 1 | 0 | 2 |
| 15 | 14 | W_07347_90_hiphop_M.jpg | 1990 | hiphop | M | 1 | 1 | 5 | 1 | 1 | 1 | 1 | 0 | 0 |
| 16 | 15 | W_16892_90_hiphop_M.jpg | 1990 | hiphop | M | 1 | 1 | 5 | 2 | 1 | 1 | 1 | 0 | 0 |
| 17 | 16 | W_16797_00_metrosexual_M.jpg | 2000 | metrosexual | M | 1 | 2 | 5 | 2 | 1 | 1 | 1 | 0 | 0 |
| 18 | 17 | W_17324_00_metrosexual_M.jpg | 2000 | metrosexual | M | 1 | 3 | 5 | 1 | 1 | 1 | 1 | 0 | 0 |
| 19 | 18 | W_17881_00_metrosexual_M.jpg | 2000 | metrosexual | M | 2 | 2 | 5 | 1 | 2 | 1 | 1 | 0 | 2 |
| 20 | 19 | W_04527_10_sportivecasual_M.jpg | 2010 | sportivecasual | M | 4 | 1 | 5 | 3 | 2 | 2 | 1 | 0 | 0 |
| 21 | 20 | W_04564_10_sportivecasual_M.jpg | 2010 | sportivecasual | M | 2 | 3 | 5 | 3 | 2 | 1 | 1 | 0 | 0 |
| 22 | 21 | W_06193_10_sportivecasual_M.jpg | 2010 | sportivecasual | M | 4 | 2 | 5 | 3 | 2 | 2 | 2 | 0 | 0 |

Figure 1. raw dataset (일부 rows, columns)

Figure 2. 이미지 샘플(W_08958_19_genderless_W)

## 1-1. 이미지 dataset 추출 및 가공

raw dataset에서 이미지에 대한 정보로 추출할 만한 요소로 참가자들의 해당 이미지에 대한 평가를 꼽았다. 이미지에 대한 요소로 '계절감, 핏, 색상, 색감, 분위기, 기타 스타일' 등이 있으며, 본 연구에서는 각각의 요소에 따라 가장 많이 선택된 특징을 해당 이미지의 특성으로 입력하였다(예를 들어 3명 중 2명이 봄 의상, 1명이 여름 의상이라고 응답했다면, 해당 의상의 계절감은 봄으로 입력된다.). 그리하여 이미지의 정보를 데이터화하여 이를 추출하였다.

### 1-1-A. 이미지 dataset 추출 및 가공 코드

```
"""// image dataset processing"""

research_dataset = pd.read_csv('/content/drive/MyDrive/학부연구생/패션/input_dataset/research_dataset.csv')
image_dataset = pd.read_csv('/content/drive/MyDrive/학부연구생/패션/input_dataset/image_dataset.csv',skiprows=1)

temp_image = image_dataset.copy()
temp_image = temp_image.rename(columns={"사용 이미지화일": "imgName", "연도": "era", "스타일": "style", "파일-성별": "gender", "노출횟수": "count"})
temp_image.head()

temp_dict = {'season':'','situation':'','fit':'','dark_bright':'','cold_warm':'','heavy_light':'',
             'cool':'','urban':'','trandy':'','stylish':'','neat':'','fancy':'','unique':'','normal':'',
             'open':'','useful':'','active':'','comfortable':'','youthful':'','feminine':'','masculine':'','soft':''}

temp_image = temp_image.append(temp_dict,ignore_index=True, sort=False)
target_list = list(temp_image.columns)[5:]
name_list = list(temp_dict.keys())
temp_image = temp_image.rename(columns={target_list[i]:name_list[i] for i in range(len(name_list))})
img_dict = {}
#answer of Q2~Q4216
q3_dict={}
#q3 (1~8) => (출근~기타) 상황에 대한 질문
```

Figure 3. 과정1. raw dataset 추출과 column명 리스트 생성

```
#research_dataset에서 img에 대한 정보를 확인.
for i in range(len(research_dataset)):
  imgName = research_dataset.loc[i,'imgName']
  #해당 이미지를 갖는 key가 없으면 [0..0]으로 초기화
  if imgName not in img_dict.keys():
    img_dict[imgName]=[0 for _ in range(23)] # index 0~21: answer , 22:count

  if imgName not in q3_dict.keys():
    q3_dict[imgName]=[0 for _ in range(9)] # index 0~7: answer, 8:count

  q2_to_q4 = research_dataset.loc[i][6:28] #q2_to_q4는 Q2~Q4216에 대한 답

  #cnt+=1
  img_dict[imgName][-1]+=1
  q3_dict[imgName][-1]+=1

  for x in range(22):
    if x ==1: #about q3 only
      select = q2_to_q4[1] #1 ~ 8
      q3_dict[imgName][select-1]+=1
    elif x==0: #about q2
      select = q2_to_q4[x] #1 or 2 or 3
      if select == 1: #봄/가을 이면 2로
        select = 2
      if select == 2: #여름이면 1로
        select = 1
      img_dict[imgName][x]+=select
    elif x==2: #about q411
      select = q2_to_q4[x] #1 or 2 or 3
      img_dict[imgName][x]+=select
    elif x==3 or x==4 or x==5: #about q412, q413, q414
      select = q2_to_q4[x] #1 or 2
      select -= 1
      img_dict[imgName][x]+=select
    else: #about other questions
      select = q2_to_q4[x] #0 or not0
      if select != 0: #only 0 or 1
        select = 1
      img_dict[imgName][x]+=select
```

Figure 4. 과정2. research dataset으로 부터 정보 추출

```
for key,values in img_dict.items():
  for i in range(22):
    if i==1: #about Q3
      num = q3_dict[key].index(max(q3_dict[key][:8]))
      num += 1 #1~8: work~etc.
      img_dict[key][1]=num

    else:
      avg = values[i]/values[-1]
      img_dict[key][i]=avg

temp_image=temp_image.set_index('imgName')
```

Figure 5. 과정3. Q3(선호도) column에 대한 정보 추출

```
for k,values in img_dict.items():
  #for season
  if values[0] < 1.5:
    temp_image.loc[k,'season']='summer'
  elif values[0] > 2.5:
    temp_image.loc[k,'season']='winter'
  else:
    temp_image.loc[k,'season']='spring_fall'
  #for situation
  s_list=[0,'work','date','event','social_gathering','daily','leisure','vacation','etc']
  temp_image.loc[k,'situation']=s_list[values[1]]
  #for fit
  if values[2] < 1.5:
    temp_image.loc[k,'fit']='loose_fit'
  elif values[2] > 1.5:
    temp_image.loc[k,'fit']='tight_fit'
  else:
    temp_image.loc[k,'fit']='suitable_fit'
  #for values[3]~values[5]
  if values[3] < 0.5:
    temp_image.loc[k,'dark_bright']='dark'
  else:
    temp_image.loc[k,'dark_bright']='bright'

  if values[4] < 0.5:
    temp_image.loc[k,'cold_warm']='cold'
  else:
    temp_image.loc[k,'cold_warm']='warm'

  if values[5] < 0.5:
    temp_image.loc[k,'heavy_light']='heavy'
  else:
    temp_image.loc[k,'heavy_light']='light'
  #for values[6]~values[21]
  s_list=[0,0,0,0,0,0,0,'cool','urban','trandy','stylish','neat','fancy',
          'unique','normal','open','useful','active','comfortable',
          'youthful','feminine','masculine','soft']
  for i in range(6,22):
    if values[i] < 0.5:
      temp_image.loc[k,s_list[i]]='not_'+s_list[i]
    else:
      temp_image.loc[k,s_list[i]]=s_list[i]
```

Figure 6. 과정4. 이미지에 대한 정보를 기반으로 새로운 dataset 생성

### 1-1-B. 이미지 dataset 추출 및 가공 결과

| imgName | era | style | gender | count | season | situation | fit | dark_bright | cold_warm | heavy_light | cool | urban | trandy |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W_00002_ | 1960 | mods | M | 1 | summer | work | tight_fit | bright | warm | light | not_cool | not_urban | not_trandy |
| W_00003_ | 1950 | ivy | M | 4 | spring_fall | work | tight_fit | dark | cold | heavy | cool | urban | not_trandy |
| W_00004_ | 1950 | ivy | M | 1 | summer | work | tight_fit | bright | warm | light | not_cool | urban | trandy |
| W_00005_ | 1960 | mods | M | 2 | spring_fall | social_gath | suitable_fit | bright | warm | light | cool | urban | not_trandy |
| W_00007_ | 1960 | mods | M | 4 | summer | event | tight_fit | bright | warm | light | not_cool | urban | trandy |
| W_00008_ | 1919 | normcore | M | 1 | winter | daily | tight_fit | bright | warm | light | cool | not_urban | not_trandy |
| W_00010_ | 1950 | ivy | M | 3 | spring_fall | work | tight_fit | dark | cold | heavy | cool | urban | not_trandy |
| W_00011_ | 1980 | bold | M | 1 | summer | social_gath | loose_fit | dark | cold | heavy | not_cool | not_urban | not_trandy |
| W_00012_ | 1950 | ivy | M | 3 | summer | daily | tight_fit | dark | cold | heavy | not_cool | not_urban | not_trandy |
| W_00013_ | 1919 | normcore | M | 1 | summer | daily | tight_fit | dark | warm | light | not_cool | urban | not_trandy |
| W_00014_ | 1960 | mods | M | 1 | summer | event | tight_fit | dark | cold | heavy | not_cool | urban | trandy |
| W_00015_ | 1950 | ivy | M | 6 | summer | work | tight_fit | bright | warm | light | cool | urban | not_trandy |
| W_00016_ | 1950 | ivy | M | 3 | summer | work | tight_fit | bright | warm | light | not_cool | not_urban | not_trandy |
| W_00017_ | 1960 | mods | M | 1 | summer | work | tight_fit | dark | warm | heavy | cool | not_urban | not_trandy |
| W_00018_ | 1960 | mods | M | 2 | spring_fall | event | tight_fit | dark | warm | heavy | cool | urban | trandy |
| W_00019_ | 1919 | normcore | M | 1 | summer | daily | tight_fit | bright | warm | heavy | cool | urban | not_trandy |
| W_00020_ | 1960 | mods | M | 1 | summer | work | tight_fit | dark | warm | light | cool | urban | not_trandy |

Figure 7. 추출한 이미지 dataset (일부 rows, columns)

## 1-2. 참가자 dataset 추출 및 가공

raw dataset에서 참가자들의 정보를 담고 있는 column을 가지고 참가자 dataset으로 추출하였다. 이때 '성별, 나이, 직업, 선호 스타일 등'을 특성으로 꼽았다.

## 1-2-A. 참가자 dataset 추출 및 가공 코드

```
"""// user dataset processing"""
user_dict = {}
#key: R_id
for i in range(len(research_dataset)):
  R_id = research_dataset.loc[i,'R_id']
  #해당 R_id를 갖는 key가 없으면 추가
  if R_id not in user_dict.keys():
    user_dict[R_id]=[0 for _ in range(10)] # 0 ~ 9 : r_gender ~ r_style5
    gender_to_style5 = research_dataset.loc[i][30:] #gender~style5에 대한 답
    for x in range(10):
      select = gender_to_style5[x]
      user_dict[R_id][x]+=select

id_list = list(user_dict.keys())
gender_list = []
age_list =[]
married_list=[]
job_list=[]
r_style1_list=[]
r_style2_list=[]
income_list=[]
r_style3_list=[]
r_style4_list=[]
r_style5_list=[]

for i in range(len(user_dict)):
  gender_list.append(list(user_dict.values())[i][0])
  age_list.append(list(user_dict.values())[i][1])
  married_list.append(list(user_dict.values())[i][2])
  job_list.append(list(user_dict.values())[i][3])
  r_style1_list.append(list(user_dict.values())[i][4])
  r_style2_list.append(list(user_dict.values())[i][5])
  income_list.append(list(user_dict.values())[i][6])
  r_style3_list.append(list(user_dict.values())[i][7])
  r_style4_list.append(list(user_dict.values())[i][8])
  r_style5_list.append(list(user_dict.values())[i][9])

temp_dict={'R_id':id_list, 'r_gender':gender_list, 'age':age_list, 'mar': married_list, 'job': job_list,
           'fancy_normal': r_style1_list,'sexual_unisexual': r_style2_list, 'income':income_list,
           'tranditional_trandy': r_style3_list,'formal_casual': r_style4_list,'active_gentle': r_style5_list}

temp_user = pd.DataFrame(temp_dict)
```

Figure 1. 과정1. raw dataset 추출과 column명 리스트 생성

```
temp_user=temp_user.set_index('R_id')
for i in range(len(temp_user)):
  if temp_user.loc[id_list[i],'r_gender']==1:
    temp_user.loc[id_list[i],'r_gender']='man'
  else:
    temp_user.loc[id_list[i],'r_gender']='woman'
  ages=[0,'20_29','30_39','40_49','50_59']
  temp_user.loc[id_list[i],'age'] = ages[temp_user.loc[id_list[i],'age']]
  if temp_user.loc[id_list[i],'mar']==1:
    temp_user.loc[id_list[i],'mar']='married'
  else:
    temp_user.loc[id_list[i],'mar']='not_married'
  jobs=[0,'housewife','tech_profession','service','office','student','etc']
  temp_user.loc[id_list[i],'job']=jobs[temp_user.loc[id_list[i],'job']]
  s1=[0,'fancy','normal']
  temp_user.loc[id_list[i],'fancy_normal']=s1[temp_user.loc[id_list[i],'fancy_normal']]
  s2=[0,'sexual','unisexual']
  temp_user.loc[id_list[i],'sexual_unisexual']=s2[temp_user.loc[id_list[i],'sexual_unisexual']]
  incomes=[0,'_200','200_300','300_400','400_500','500_600','600_']
  temp_user.loc[id_list[i],'income']=incomes[temp_user.loc[id_list[i],'income']]
  s3=[0,'tranditional','trandy']
  temp_user.loc[id_list[i],'tranditional_trandy']=s3[temp_user.loc[id_list[i],'tranditional_trandy']]
  s4=[0,'formal','casual']
  temp_user.loc[id_list[i],'formal_casual']=s4[temp_user.loc[id_list[i],'formal_casual']]
  s5=[0,'active','gentle']
  temp_user.loc[id_list[i],'active_gentle']=s5[temp_user.loc[id_list[i],'active_gentle']]

temp_user.to_csv('/content/drive/MyDrive/학부연구생/패션/output_dataset/user_preprocessing_out.csv')
```

Figure 2. 과정2. research dataset으로 부터 정보 추출 및 새로운 dataset 생성

## 1-2-B. 참가자 dataset 추출 및 가공 결과

| R_id | r_gender | age | mar | job | fancy_norm | sexual_uni | income | tranditional | formal_cas | active_ger |
|---|---|---|---|---|---|---|---|---|---|---|
| 27 | man | 50_59 | not_marrie | office | fancy | unisexual | 600_ | trandy | casual | active |
| 133 | man | 30_39 | married | etc | normal | sexual | _200 | trandy | casual | gentle |
| 179 | woman | 40_49 | married | office | normal | sexual | 500_600 | tranditional | casual | gentle |
| 289 | woman | 30_39 | not_marrie | housewife | normal | unisexual | 300_400 | tranditional | casual | active |
| 1022 | woman | 30_39 | married | etc | normal | unisexual | _200 | tranditional | casual | gentle |
| 1027 | woman | 20_29 | married | etc | normal | unisexual | _200 | trandy | casual | gentle |
| 1073 | woman | 30_39 | not_marrie | housewife | normal | unisexual | 300_400 | trandy | casual | active |
| 1512 | woman | 40_49 | not_marrie | housewife | normal | unisexual | 500_600 | tranditional | casual | gentle |
| 1513 | woman | 40_49 | not_marrie | housewife | normal | sexual | _200 | tranditional | casual | gentle |
| 1516 | woman | 40_49 | not_marrie | housewife | normal | unisexual | 200_300 | trandy | casual | gentle |
| 1880 | woman | 30_39 | married | etc | normal | unisexual | 300_400 | trandy | casual | active |
| 1968 | woman | 50_59 | not_marrie | tech_profe | normal | unisexual | 200_300 | trandy | casual | gentle |
| 1969 | woman | 50_59 | not_marrie | service | normal | unisexual | _200 | trandy | casual | gentle |
| 1970 | woman | 40_49 | not_marrie | housewife | normal | unisexual | 600_ | trandy | casual | active |

Figure 3. 추출한 참가자 dataset (일부 rows, columns)

## 1-3. research dataset 가공

raw dataset에서 추출 및 가공한 이미지 dataset과 참가자 dataset을 가지고 research dataset을 가공함으로써 머신러닝 모델을 돌리기 위한 dataset을 만들어 준다.

## 1-3-A. reserach dataset 가공 코드

```
"""//research dataset preprocessing"""
temp_research = research_dataset.copy()
temp_research=temp_research.drop(['Q1'],axis=1)
columns_list=list(temp_research.columns)
temp_dict = {'season':'','situation':'','fit':'','dark_bright':'','cold_warm':'','heavy_light':'',
             'cool':'','urban':'','trandy':'','stylish':'','neat':'','fancy':'','unique':'','normal':'',
             'open':'','useful':'','active':'','comfortable':'','youthful':'','feminine':'','masculine':'','soft':''}
rename_list=list(temp_dict.keys())
print(len(rename_list),rename_list)
print(len(columns_list),columns_list)

columns_list=columns_list[5:]
print(len(rename_list),rename_list)
print(len(columns_list),columns_list)

temp_research = temp_research.rename(columns={columns_list[i]: rename_list[i] for i in range(len(rename_list))} )
temp_research.head()

temp_dict={'R_id':id_list, 'r_gender':gender_list, 'age':age_list, 'mar': married_list, 'job': job_list,
           'fancy_normal': r_style1_list,'sexual_unisexual': r_style2_list, 'income':income_list,
           'tranditional_trandy': r_style3_list,'formal_casual': r_style4_list,'active_gentle': r_style5_list}

rename_list=list(temp_dict.keys())
columns_list=list(temp_research.columns)

columns_list=columns_list[33:]
rename_list=rename_list[5:]
temp_research = temp_research.rename(columns={columns_list[i]: rename_list[i] for i in range(len(rename_list))} )


#research_dataset[6:28]는 image_dataset에 mapping by ImgName
#research_dataset[29:]은 user_dataset에 mapping by R_id
image_name_list=list(temp_image.index)
image_name_list.pop() #last row -> nan. don't know the reaseon.
image_dict={image_name_list[i]: list(temp_image.loc[image_name_list[i]])[4:] for i in range(len(image_name_list))}
user_id_list=list(temp_user.index)
user_dict={user_id_list[i]: list(temp_user.loc[user_id_list[i]]) for i in range(len(user_id_list))}
```

Figure 4. 과정1. research dataset의 column명에 맞는 리스트 생성 등

```
#Q5(선호여부 정보 object화)

for i in range(len(temp_research)):
  if temp_research.loc[i,'Q5'] == 2:
    temp_research.loc[i,'Q5']='Positive'
  else:
    temp_research.loc[i,'Q5']='Negative'

#img정보 처리

temp_research = temp_research.set_index('imgName')
temp_dict = {'season':'','situation':'','fit':'','dark_bright':'','cold_warm':'','heavy_light':'',
             'cool':'','urban':'','trandy':'','stylish':'','neat':'','fancy':'','unique':'','normal':'',
             'open':'','useful':'','active':'','comfortable':'','youthful':'','feminine':'','masculine':'','soft':''}
column_list=list(temp_dict.keys())

for i in range(len(image_name_list)):
  image_name = image_name_list[i]
  for j in range(len(list(image_dict.values())[0])):
    col = column_list[j]
    temp_research.loc[image_name, col]=image_dict[image_name][j]

#user 정보 처리

temp_research = temp_research.set_index('R_id')
temp_dict={'R_id':id_list, 'r_gender':gender_list, 'age':age_list, 'mar': married_list, 'job': job_list,
           'fancy_normal': r_style1_list,'sexual_unisexual': r_style2_list, 'income':income_list,
           'tranditional_trandy': r_style3_list,'formal_casual': r_style4_list,'active_gentle': r_style5_list}
column_list=list(temp_dict.keys())
column_list.pop(0)

for i in range(len(user_id_list)):
  user_id = user_id_list[i]
  for j in range(len(list(user_dict.values())[0])):
    col = column_list[j]
    temp_research.loc[user_id, col]=user_dict[user_id][j]

temp_research=temp_research.set_index('E_id')

temp_research.to_csv('/content/drive/MyDrive/학부연구생/패션/output_dataset/research_preprocessing_out.csv')
```

Figure 5. 과정2. 기존의 dataset을 이용하여 새로운 research dataset 생성

## 1-3-B. research dataset 가공 결과

| E_id | era | style | gender | season | situation | fit | dark_brig | cold_warm | heavy_ligh | cool | urban | trandy | stylish | neat | fancy | unique | normal | open | useful | active | comfortab | youthful | feminine | masculine | soft |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1950 | ivy | M | spring_fall | work | tight_fit | bright | warm | light | not_cool | not_urban | trandy | not_stylish | not_neat | not_fancy | not_uniqu | not_norm | open | not_useful | active | comfortab | not_youth | not_femin | not_masc | not_soft |
| 2 | 1950 | ivy | M | spring_fall | daily | tight_fit | bright | warm | light | not_cool | not_urban | trandy | not_stylish | not_neat | not_fancy | not_uniqu | not_norm | not_open | not_useful | not_active | comfortab | not_youth | not_femin | not_masc | not_soft |
| 3 | 1950 | ivy | M | summer | date | suitable_fi | dark | warm | light | not_cool | urban | trandy | not_stylish | not_neat | fancy | unique | not_norm | open | not_useful | not_active | not_comfc | not_youth | not_femin | masculine | not_soft |
| 4 | 1960 | mods | M | summer | work | tight_fit | bright | warm | light | cool | urban | trandy | not_stylish | neat | not_fancy | not_uniqu | not_norm | not_open | useful | not_active | comfortab | not_youth | not_femin | not_masc | soft |
| 5 | 1960 | mods | M | spring_fall | date | tight_fit | dark | cold | heavy | cool | not_urban | not_trandy | stylish | not_neat | not_fancy | not_uniqu | normal | not_open | useful | active | not_comfc | not_youth | not_femin | masculine | not_soft |
| 6 | 1960 | mods | M | summer | date | tight_fit | dark | cold | heavy | not_cool | urban | not_trandy | not_stylish | not_neat | not_fancy | not_uniqu | not_norm | not_open | not_useful | not_active | not_comfc | not_youth | not_femin | not_masc | not_soft |
| 7 | 1970 | hippie | M | summer | social_gat | tight_fit | bright | cold | heavy | not_cool | not_urban | not_trandy | not_stylish | not_neat | not_fancy | not_uniqu | not_norm | open | not_useful | not_active | not_comfc | not_youth | not_femin | not_masc | not_soft |
| 8 | 1970 | hippie | M | summer | daily | tight_fit | bright | warm | light | not_cool | urban | trandy | stylish | not_neat | not_fancy | not_uniqu | not_norm | not_open | useful | active | not_comfc | youthful | not_femin | not_masc | not_soft |
| 9 | 1970 | hippie | M | summer | social_gat | tight_fit | bright | warm | light | not_cool | not_urban | not_trandy | not_stylish | not_neat | not_fancy | unique | not_norm | not_open | not_useful | not_active | not_comfc | not_youth | not_femin | not_masc | not_soft |
| 10 | 1980 | bold | M | summer | work | loose_fit | dark | cold | heavy | not_cool | not_urban | trandy | not_stylish | not_neat | not_fancy | not_uniqu | not_norm | not_open | not_useful | not_active | not_comfc | not_youth | not_femin | not_masc | not_soft |
| 11 | 1980 | bold | M | spring_fall | work | tight_fit | dark | cold | heavy | cool | not_urban | not_trandy | not_stylish | not_neat | not_fancy | unique | not_norm | open | not_useful | not_active | not_comfc | not_youth | not_femin | not_masc | not_soft |
| 12 | 1980 | bold | M | spring_fall | date | tight_fit | dark | cold | heavy | not_cool | not_urban | not_trandy | not_stylish | not_neat | not_fancy | not_uniqu | normal | not_open | not_useful | not_active | not_comfc | not_youth | not_femin | masculine | not_soft |
| 13 | 1990 | hiphop | M | summer | daily | suitable_fi | bright | cold | light | not_cool | urban | trandy | not_stylish | not_neat | fancy | not_uniqu | not_norm | not_open | not_useful | active | comfortab | not_youth | not_femin | not_masc | not_soft |
| 14 | 1990 | hiphop | M | summer | daily | loose_fit | dark | cold | heavy | not_cool | not_urban | not_trandy | not_stylish | not_neat | not_fancy | not_uniqu | not_norm | open | not_useful | active | comfortab | not_youth | not_femin | not_masc | soft |
| 15 | 1990 | hiphop | M | summer | daily | tight_fit | dark | cold | heavy | not_cool | not_urban | not_trandy | stylish | not_neat | not_fancy | unique | not_norm | not_open | not_useful | not_active | not_comfc | not_youth | not_femin | not_masc | not_soft |
| 16 | 2000 | metrosexu | M | summer | daily | tight_fit | dark | cold | light | not_cool | not_urban | not_trandy | not_stylish | not_neat | not_fancy | not_uniqu | not_norm | open | not_useful | not_active | not_comfc | not_youth | not_femin | masculine | not_soft |
| 17 | 2000 | metrosexu | M | winter | daily | loose_fit | dark | cold | heavy | not_cool | not_urban | not_trandy | not_stylish | not_neat | not_fancy | not_uniqu | not_norm | not_open | not_useful | not_active | comfortab | not_youth | not_femin | not_masc | not_soft |
| 18 | 2000 | metrosexu | M | summer | daily | loose_fit | bright | cold | heavy | not_cool | urban | trandy | not_stylish | not_neat | not_fancy | not_uniqu | not_norm | not_open | not_useful | not_active | comfortab | not_youth | not_femin | not_masc | not_soft |
| 19 | 2010 | sportiveca | M | summer | daily | tight_fit | bright | warm | heavy | not_cool | not_urban | not_trandy | not_stylish | not_neat | not_fancy | not_uniqu | normal | open | not_useful | not_active | comfortab | youthful | not_femin | not_masc | not_soft |
| 20 | 2010 | sportiveca | M | winter | daily | tight_fit | bright | cold | heavy | not_cool | not_urban | not_trandy | not_stylish | not_neat | not_fancy | not_uniqu | normal | not_open | not_useful | not_active | not_comfc | not_youth | not_femin | not_masc | not_soft |
| 21 | 2010 | sportiveca | M | summer | daily | tight_fit | bright | warm | light | not_cool | not_urban | trandy | stylish | neat | not_fancy | not_uniqu | not_norm | not_open | useful | active | not_comfc | not_youth | not_femin | not_masc | not_soft |
| 22 | 2010 | sportiveca | M | summer | daily | tight_fit | dark | cold | light | not_cool | not_urban | not_trandy | not_stylish | not_neat | not_fancy | not_uniqu | not_norm | not_open | not_useful | active | comfortab | not_youth | not_femin | not_masc | not_soft |
| 23 | 2019 | normcore | M | summer | work | tight_fit | bright | warm | light | not_cool | not_urban | trandy | stylish | neat | fancy | not_uniqu | not_norm | not_open | not_useful | not_active | not_comfc | not_youth | feminine | not_masc | not_soft |
| 24 | 2019 | normcore | M | summer | daily | suitable_fi | bright | warm | light | cool | not_urban | trandy | stylish | not_neat | not_fancy | not_uniqu | normal | not_open | useful | not_active | comfortab | not_youth | not_femin | not_masc | not_soft |
| 25 | 2019 | normcore | M | summer | social_gat | suitable_fi | dark | cold | light | not_cool | not_urban | trandy | not_stylish | not_neat | not_fancy | unique | not_norm | open | not_useful | not_active | not_comfc | not_youth | not_femin | not_masc | not_soft |
| 51 | 1950 | ivy | M | summer | work | tight_fit | bright | warm | light | cool | urban | trandy | stylish | neat | not_fancy | not_uniqu | not_norm | not_open | not_useful | not_active | not_comfc | not_youth | not_femin | not_masc | not_soft |
| 52 | 1950 | ivy | M | summer | work | loose_fit | bright | warm | light | not_cool | not_urban | trandy | not_stylish | neat | not_fancy | not_uniqu | not_norm | not_open | not_useful | not_active | comfortab | not_youth | not_femin | not_masc | not_soft |
| 53 | 1950 | ivy | M | summer | date | suitable_fi | dark | warm | light | not_cool | urban | trandy | not_stylish | not_neat | fancy | unique | not_norm | open | not_useful | not_active | not_comfc | not_youth | not_femin | masculine | not_soft |
| 54 | 1960 | mods | M | summer | work | tight_fit | dark | warm | light | not_cool | not_urban | not_trandy | not_stylish | neat | not_fancy | not_uniqu | normal | not_open | useful | active | comfortab | not_youth | not_femin | not_masc | not_soft |
| 55 | 1960 | mods | M | spring_fall | social_gat | loose_fit | bright | warm | heavy | not_cool | not_urban | not_trandy | not_stylish | not_neat | not_fancy | not_uniqu | not_norm | not_open | not_useful | not_active | not_comfc | not_youth | feminine | not_masc | not_soft |
| 56 | 1960 | mods | M | spring_fall | social_gat | tight_fit | dark | warm | light | not_cool | urban | trandy | stylish | neat | not_fancy | not_uniqu | not_norm | not_open | not_useful | not_active | not_comfc | not_youth | not_femin | masculine | not_soft |
| 57 | 1970 | hippie | M | summer | social_gat | tight_fit | dark | cold | heavy | not_cool | not_urban | not_trandy | not_stylish | not_neat | not_fancy | unique | not_norm | not_open | not_useful | not_active | not_comfc | not_youth | not_femin | not_masc | not_soft |
| 58 | 1970 | hippie | M | summer | social_gat | loose_fit | bright | warm | light | not_cool | not_urban | trandy | stylish | not_neat | not_fancy | not_uniqu | not_norm | not_open | not_useful | not_active | not_comfc | youthful | not_femin | not_masc | not_soft |

Figure 6. 추출한 research dataset (일부 rows, columns)

## 2. 모델링

전처리된 dataset을 가지고 머신러닝 모델 LogisticRegression, AdaBoostClassifier, GradientBoostingClassifier를 이용하여 각각의 정확도를 확인하였다. 이때 각 model에 해당 dataset을 적용하기 위해 dataset을 그대로 적용하지 않고 one hot encoding 방법으로 적용하였다. 또한, 하이퍼 파라미터를 입력하지 않고 도출된 결과와 하이퍼 파라미터를 튜닝하여 도출된 결과를 비교할 수 있도록 구축하였다.

## 2-1-A. 모델링 코드

```
import ...

research_dataset = pd.read_csv('/content/drive/MyDrive/학부연구생/패션/output_dataset/research_preprocessing_out.csv')
df = research_dataset.copy()
df = df.set_index('E_id')
# feature와 label 나누기
X = df.drop(['Q5'], axis=1)
y = df['Q5']
# one hot encoding
X = pd.get_dummies(X)
# split X and y into traning and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
#모델 종류
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier

#default 학습 결과
before_result = {'logistic':0, 'abc': 0, 'gb': 0}
logistic_model = LogisticRegression()
logistic_model.fit(X_train,y_train)
logistic_y_pred = logistic_model.predict(X_test)
# LogisticRegression Accuracy
print("Logistic Regression Accuracy:",metrics.accuracy_score(y_test, logistic_y_pred))
before_result['logistic']=metrics.accuracy_score(y_test, logistic_y_pred)

abc_model = AdaBoostClassifier()
abc_model.fit(X_train,y_train)
abc_y_pred = abc_model.predict(X_test)
# AdaBoostClassifier Accuracy
print("AdaBoost Classifier Accuracy:",metrics.accuracy_score(y_test, abc_y_pred))
before_result['abc']=metrics.accuracy_score(y_test, abc_y_pred)

gb_model = GradientBoostingClassifier()
gb_model.fit(X_train, y_train, sample_weight=None, monitor=None)
gb_y_pred = gb_model.predict(X_test)
# GradientBoostingClassifier Accuracy
print("GradientBoosting Classifier Accuracy:",metrics.accuracy_score(y_test, gb_y_pred))
before_result['gb']=metrics.accuracy_score(y_test, gb_y_pred)
```

Figure 1. 과정1. dataset을 로드한 후, 하이퍼파라미터를 적용하지 않고 모델 적용

```
## logistic regression 하이퍼파라미터 튜닝용 함수
def logistic_tuning(X_train, y_train, params):
    #model = LogisticRegression(random_state = 99)
    model = LogisticRegression()
    # 파라미터 튜닝(train data 전체를 넣어서 5-fold cv)
    grid = GridSearchCV(model, params, scoring = 'roc_auc', cv = 5)
    grid.fit(X_train, y_train)
    print(grid.best_params_)
    print(grid.best_score_)
    return grid.best_estimator_
logistic_param = {'C':[(4.0+x/10.0) for x in range(20)], 'max_iter': [150+m for m in range(0,110,10)]}
logistic_tuning_result = logistic_tuning(X_train, y_train, params = logistic_param) #최적의 하이퍼파라미터 확인

## AdaBoost Classifier 하이퍼파라미터 튜닝용 함수
def abc_tuning(train_sprs, y, params):
    model = AdaBoostClassifier()
    # 파라미터 튜닝(train data 전체를 넣어서 5-fold cv)
    grid = GridSearchCV(model, params, scoring = 'roc_auc', cv = 5)
    grid.fit(train_sprs, y)
    print(grid.best_params_)
    print(grid.best_score_)
    return grid.best_estimator_
abc_param = {'n_estimators':[50+n for n in range(0,100,10)],'learning_rate':[1.0 + 0.005*i for i in range(1,11)]}
abc_tuning_result = abc_tuning(X_train, y_train, params = abc_param) #최적의 하이퍼파라미터 확인.

## gradient boosting 하이퍼파라미터 튜닝용 함수
def gb_tuning(train_sprs, y, params):
    model = GradientBoostingClassifier()
    # 파라미터 튜닝(train data 전체를 넣어서 5-fold cv)
    grid = GridSearchCV(model, params, scoring = 'roc_auc', cv = 5)
    grid.fit(train_sprs, y)
    print(grid.best_params_)
    print(grid.best_score_)
    return grid.best_estimator_
gb_param = {'n_estimators': [100+n for n in range(10,100,10)], 'max_depth': [3+m for m in range(1,6)]}
gb_tuning_result = gb_tuning(X_train, y_train, params = gb_param) #최적의 하이퍼파라미터 확인.
```

Figure 2. 과정2. 각각의 머신러닝 모델에 맞게 하이퍼파라미터 튜닝함수 적용

```
#최적의 하이퍼파라미터로 학습한 후 결과
after_result = {'logistic':0, 'abc': 0, 'gb': 0}

#{'C': 5.9, 'max_iter': 250}
logistic_model = LogisticRegression(C=5.9, max_iter=250)
logistic_model.fit(X_train,y_train)
logistic_y_pred = logistic_model.predict(X_test)
# Model Accuracy, how often is the classifier correct?
print("Logistic Regression Accuracy:",metrics.accuracy_score(y_test, logistic_y_pred))
after_result['logistic'] = metrics.accuracy_score(y_test, logistic_y_pred)

#{'learning_rate': 1.05, 'n_estimators': 100}
abc_model = AdaBoostClassifier(learning_rate=1.05, n_estimators=100)
abc_model.fit(X_train,y_train)
abc_y_pred = abc_model.predict(X_test)
# Model Accuracy
print("AdaBoost Classifier Accuracy:",metrics.accuracy_score(y_test, abc_y_pred))
after_result['abc'] = metrics.accuracy_score(y_test, abc_y_pred)

#{'max_depth': 5, 'n_estimators': 190}
gb_model = GradientBoostingClassifier(max_depth=5, n_estimators=190)
gb_model.fit(X_train, y_train, sample_weight=None, monitor=None)
gb_y_pred = gb_model.predict(X_test)
#Model Accuracy
print("GradientBoosting Classifier Accuracy:",metrics.accuracy_score(y_test, gb_y_pred))
after_result['gb'] = metrics.accuracy_score(y_test, gb_y_pred)
```

Figure 3. 과정3. 하이퍼파라미터 튜닝 후의 결과 확인

```
#plot of before_result
names = list(before_result.keys())
values=[]
for name in names:
  values.append(before_result[name])

plt.figure(figsize=(3, 3))

plt.scatter(names, values)
plt.title('default_result')
#plt.show()
print(before_result)

plt.savefig('/content/drive/MyDrive/학부연구생/패션/output_image/default_result.png')
plt.close()

#plot of after_result
names = list(after_result.keys())
values=[]
for name in names:
  values.append(after_result[name])

plt.figure(figsize=(3, 3))

plt.scatter(names, values)
plt.title('tuned_result')
#plt.show()
print(after_result)
plt.savefig('/content/drive/MyDrive/학부연구생/패션/output_image/tuned_result.png')
plt.close()

#파라미터 확인
logistic_model.get_params()
abc_model.get_params()
gb_model.get_params()
```

Figure 4. 과정4. 결과를 그래프로 도출

## 3. 결과

| Model | Default Acc. | Tuned Acc. |
|---|---|---|
| LogisticRegression | 0.7235209 | 0.7280919 |
| AdaBoostClassifier | 0.7284837 | 0.7287449 |
| GradientBoostingClassifier | 0.7269165 | 0.7403682 |

Table 1. A table caption.

머신러닝 모델 LogisticRegression, AdaBoostClassifier, GradientBoostingClassifier에 동일한 dataset을 적용한 결과이다. 먼저, LogisticRegression의 경우 default accuracy는 0.7235209, 튜닝 후 accuracy는 0.7280919이다. 하이퍼 파라미터를 튜닝한 결과, 'C': 5.9, 'max_iter': 250이 최적의 파라미터로 확인되었다. 다음으로 AdaBoostClassifier의 경우 default accracy는 0.7284837, 튜닝 후 accuracy는 0.7287449이며, 튜닝한 최종 하이퍼 파라미터는 'learning_rate': 1.05, 'n_estimators': 100이다. 마찬가지로 GradientBoostingClassifier의 경우도 default accracy는 0.7269165, 튜닝 후 accuracy는 0.7403682, 튜닝한 최종 하이퍼 파라미터는 'max_depth': 5, 'n_estimators': 190으로 확인할 수 있었다.

## 3-1. 적용된 최적의 파라미터

- **LogisticRegression** 'C': 5.9, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_ratio': None, 'max_iter': 250, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'l2', 'random_state': None, 'solver': 'lbfgs', 'tol': 0.0001, 'verbose': 0, 'warm_start': False
- **AdaBoostClassifier** 'algorithm': 'SAMME.R', 'base_estimator': None, 'learning_rate': 1.05, 'n_estimators': 100, 'random_state': None
- **GradientBoostingClassifier** 'ccp_alpha': 0.0, 'criterion': 'friedman_mse', 'init': None, 'learning_rate': 0.1, 'loss': 'deviance', 'max_depth': 5, 'max_features': None, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 190, 'n_iter_no_change': None, 'random_state': None, 'subsample': 1.0, 'tol': 0.0001, 'validation_fraction': 0.1, 'verbose': 0, 'warm_start': False

## 3-2. 그래프

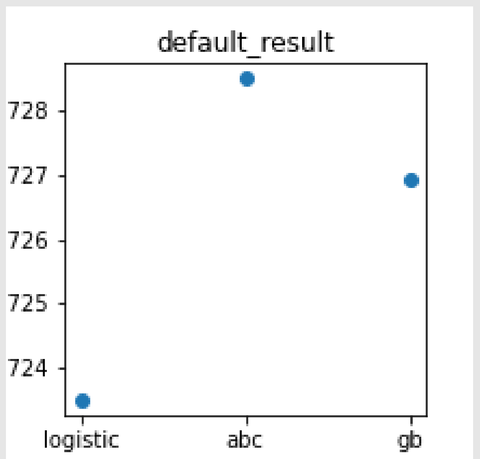


Figure 5. Default Accuracy

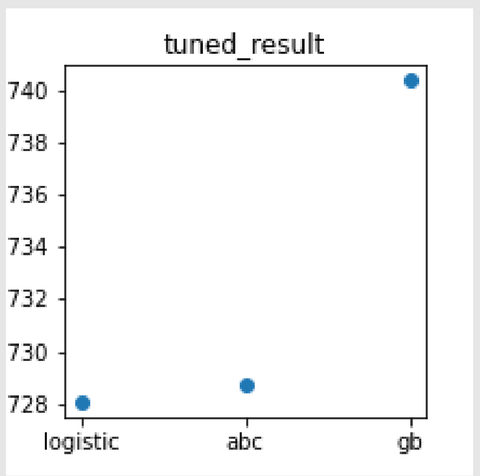


Figure 6. Tuned Accuracy