
빅데이터 분석을 통한
고객 반응 예측 모델 개발

# Contents

# 01 Dataset
# 데이터 소개

## 사용 데이터 – Customer Personality Analysis (https://www.kaggle.com/imakash3011/customer-personality-analysis)

**Instance** : (2240 x 29) = 64,960 개

**File Type** : CSV

**Columns** :

ID: 고객ID
Year_Birth: 고객 생년월일
Education: 고객의 교육수준
Marital_Status: 고객의 결혼여부
Income: 고객의 연간 가구소득
Kidhome: 고객의 가구 내 자녀의 수
Teenhome: 고객의 가구 내 청소년 수
Dt_Customer: 고객이 회사에 등록한 날짜(회원가입 일자)
Recency: 고객의 마지막 구매 후 지난 일수
Complain: 지난 2년동안 고객 불만 제기한 경우 1, 그렇지 않을 경우 0

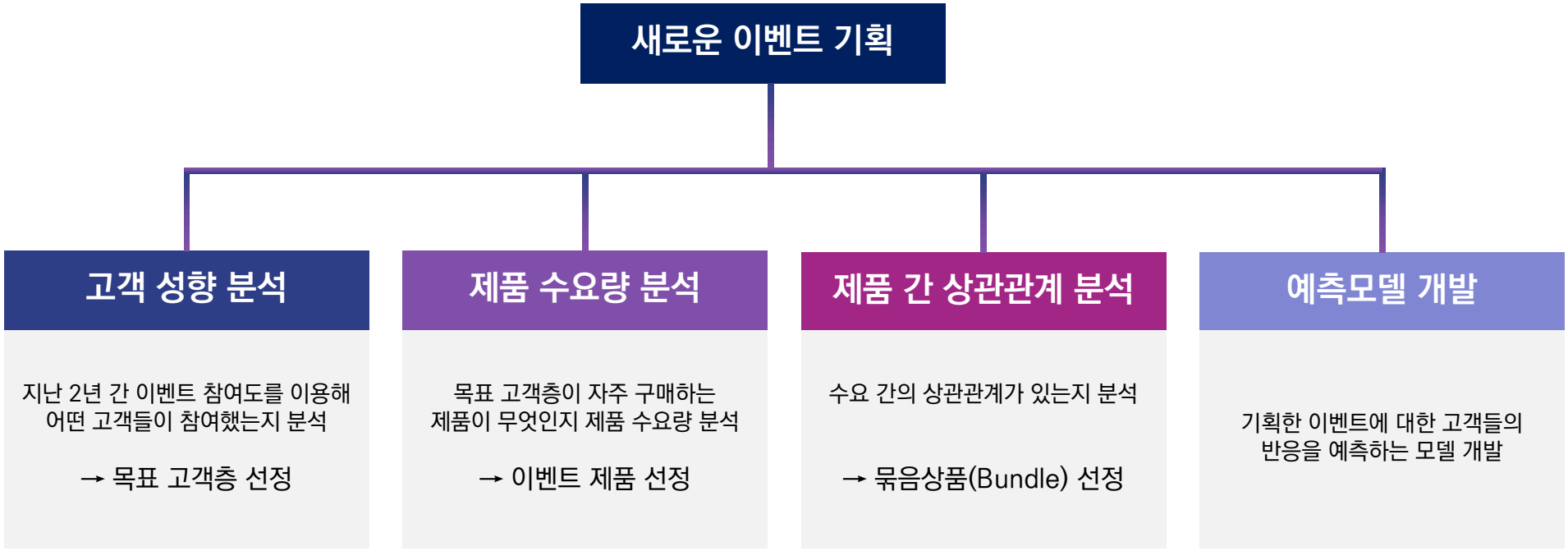
MntWines: 지난 2년 간 와인 소비량
MntFruits: 지난 2년 간 과일 소비량
MntMeatProducts: 지난 2년 간 육류 소비량
MntFishProducts: 지난 2년 간 생선 소비량
MntSweetProducts: 지난 2년 간 간식 소비량
MntGoldProds: 지난 2년 간 금 소비량

NumDealsPurchases: 할인제품 구매 건 수
AcceptedCmp1: 첫 번째 할인 이벤트에서 고객이 구매한 경우 1, 그렇지 않은 경우 0
AcceptedCmp2: 두 번째 할인 이벤트에서 고객이 구매한 경우 1, 그렇지 않은 경우 0
AcceptedCmp3: 세 번째 할인 이벤트에서 고객이 구매한 경우 1, 그렇지 않은 경우 0
AcceptedCmp4: 네 번째 할인 이벤트에서 고객이 구매한 경우 1, 그렇지 않은 경우 0
AcceptedCmp5: 다섯 번째 할인 이벤트에서 고객이 구매한 경우 1, 그렇지 않은 경우 0
Response: 지난 이벤트에서 고객이 구매한 경우 1, 그렇지 않은 경우 0

NumWebPurchases: 웹사이트를 통한 구매 건 수
NumCatalogPurchases: 카탈로그를 통한 구매 건 수
NumStorePurchases: 매장에서 직접 구매한 건 수
NumWebVisitsMonth: 지난 달 회사 웹 사이트 방문 수

# 02 Motivation
# 과제 목표

기업에서 새로운 이벤트 진행을 위해 이벤트를 기획 할 때, 데이터 분석을 통해 **목표고객층, 묶음판매 제품 의사결정**에 도움을 주고 행사에 대한 **고객 반응 예측**을 할 수 있도록 하기 위해 본 과제를 진행



# 03 Preprocessing 데이터 전처리

## Step1. 결측치 제거

```
data.isna().sum()
```

```
ID                   0
Year_Birth           0
Education            0
Marital_Status       0
Income              24
Kidhome              0
Teenhome             0
Dt_Customer          0
Recency              0
MntWines             0
MntFruits            0
MntMeatProducts      0
MntFishProducts      0
MntSweetProducts     0
MntGoldProds         0
NumDealsPurchases    0
NumWebPurchases      0
NumCatalogPurchases  0
NumStorePurchases    0
NumWebVisitsMonth    0
AcceptedCmp3         0
AcceptedCmp4         0
AcceptedCmp5         0
AcceptedCmp1         0
AcceptedCmp2         0
Complain             0
Z_CostContact        0
Z_Revenue            0
Response             0
dtype: int64
```

```
data = data.dropna(axis=0)
```

24개의 결측치 제거

```
data.isna().sum()
```

```
ID                   0
Year_Birth           0
Education            0
Marital_Status       0
Income               0
Kidhome              0
Teenhome             0
Dt_Customer          0
Recency              0
MntWines             0
MntFruits            0
MntMeatProducts      0
MntFishProducts      0
MntSweetProducts     0
MntGoldProds         0
NumDealsPurchases    0
NumWebPurchases      0
NumCatalogPurchases  0
NumStorePurchases    0
NumWebVisitsMonth    0
AcceptedCmp3         0
AcceptedCmp4         0
AcceptedCmp5         0
AcceptedCmp1         0
AcceptedCmp2         0
Complain             0
Z_CostContact        0
Z_Revenue            0
Response             0
dtype: int64
```

## Step2. 컬럼 값 정리

```
pd.DataFrame(data.nunique()).sort_values(0)
# 각 컬럼이 몇개의 값을 갖고있는지 확인
# Z_Revenue 와 Z_CostContact는 하나의 값만
# 가지고 있으므로 분석에 필요없는 컬럼. --> 삭제
```

| | 0 |
|---|---|
| Z_Revenue | 1 |
| Z_CostContact | 1 |
| Response | 2 |

사용하지 않는 컬럼 삭제

```
data = data.drop(['Z_Revenue','Z_CostContact'],axis = 1)
data = data.drop(['Dt_Customer','ID'],axis = 1)
```

```
data['Age'] = 2021- data.Year_Birth + 1
data = data.drop('Year_Birth',axis = 1)
```

출생년도를 이용해 Age 컬럼 생성

```
data.head()
```

| | Education | Marital_Status | Income | Kidhome | Age |
|---|---|---|---|---|---|
| 0 | Graduation | Single | 58138.0 | 0 | 65 |
| 1 | Graduation | Single | 46344.0 | 1 | 68 |
| 2 | Graduation | Together | 71613.0 | 0 | 57 |
| 3 | Graduation | Together | 26646.0 | 1 | 38 |
| 4 | PhD | Married | 58293.0 | 1 | 41 |

# 03 Preprocessing 데이터 전처리

## Step2. 컬럼 값 정리(계속)

```
data['Education'].value_counts()
# Graduation = 학사
# Master = 석사
# 2n Cycle = 석사
# PhD = 박사
# Basic = 고졸
# Master 와 2n Cycle 모두 석사인데 미국과 유럽의 표기 차이이기 때문에 Master로 통합
# Graduation은 Bachelor으로 용어 변경
```

```
Graduation    1116
PhD            481
Master         365
2n Cycle       200
Basic           54
Name: Education, dtype: int64
```

```
data['Education'] = data['Education'].replace(['2n Cycle', 'Graduation'], ['Master', 'Bachelor'])
data['Education'].value_counts()
```

```
Bachelor    1116
Master       565
PhD          481
Basic         54
Name: Education, dtype: int64
```

'Education'컬럼의 표기법이 다른 데이터 값을
Basic(고졸) / Bachelor(학사) / Master(석사) / PhD(박사)
4가지 값으로 통합 및 변환

```
data['Marital_Status'].value_counts()
# 결혼 여부도 분석용이함을 위해 기혼/미혼/이혼으로 재설정
```

```
Married     857
Together    573
Single      471
Divorced    232
Widow        76
Alone         3
Absurd        2
YOLO          2
Name: Marital_Status, dtype: int64
```

```
data['Marital_Status'] = data['Marital_Status'].replace(['Alone','YOLO','Absurd'],'Single')
data['Marital_Status'] = data['Marital_Status'].replace('Together', 'Married')
data['Marital_Status'] = data['Marital_Status'].replace('Widow', 'Divorced')
data['Marital_Status'].value_counts()
```

```
Married     1430
Single       478
Divorced     308
Name: Marital_Status, dtype: int64
```

'Marital_Status'컬럼도 표기법이 다른 데이터 값들을
Married(기혼) / Single(미혼) / Divorced(이혼)
3가지 값으로 통합 및 변환

# 03 Preprocessing 데이터 전처리

## Step2. 컬럼 값 정리(계속)

```
# AcceptedCmp1: 첫 번째 할인 이벤트에서 고객이 구매한 경우 1, 그렇지 않은 경우 0
# AcceptedCmp2: 두 번째 할인 이벤트에서 고객이 구매한 경우 1, 그렇지 않은 경우 0
# AcceptedCmp3: 세 번째 할인 이벤트에서 고객이 구매한 경우 1, 그렇지 않은 경우 0
# AcceptedCmp4: 네 번째 할인 이벤트에서 고객이 구매한 경우 1, 그렇지 않은 경우 0
# AcceptedCmp5: 다섯 번째 할인 이벤트에서 고객이 구매한 경우 1, 그렇지 않은 경우 0
# Response: 지난 이벤트에서 고객이 구매한 경우 1, 그렇지 않은 경우 0
# 위 컬럼들의 경우 각 이벤트(Cmp1,2,3,4,5)를 집계한 값이 Response이므로 삭제를 해준다.
data = data.drop(['AcceptedCmp1','AcceptedCmp2','AcceptedCmp3','AcceptedCmp4','AcceptedCmp5'],axis = 1)
data.info()
```

지난 2년간 시행한 이벤트 5개의 참여 여부를 보여주는 “AcceptedCmp1~5”의 컬럼은 “Response” 컬럼에서 집계되어 있기 때문에 필요 없다고 판단 → 삭제

```
print(data.info())
print(data.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2205 entries, 0 to 2239
Data columns (total 20 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Education            2205 non-null   object
 1   Marital_Status       2205 non-null   object
 2   Income               2205 non-null   float64
 3   Kidhome              2205 non-null   int64
 4   Teenhome             2205 non-null   int64
 5   Recency              2205 non-null   int64
 6   MntWines             2205 non-null   int64
 7   MntFruits            2205 non-null   int64
 8   MntMeatProducts      2205 non-null   int64
 9   MntFishProducts      2205 non-null   int64
 10  MntSweetProducts     2205 non-null   int64
 11  MntGoldProds         2205 non-null   int64
 12  NumDealsPurchases    2205 non-null   int64
 13  NumWebPurchases      2205 non-null   int64
 14  NumCatalogPurchases  2205 non-null   int64
 15  NumStorePurchases    2205 non-null   int64
 16  NumWebVisitsMonth    2205 non-null   int64
 17  Complain             2205 non-null   int64
 18  Response             2205 non-null   int64
 19  Age                  2205 non-null   int64
dtypes: float64(1), int64(17), object(2)
memory usage: 361.8+ KB
None
(2205, 20)
```

컬럼이 정리된 데이터 정보

Columns : 29개 → 20개

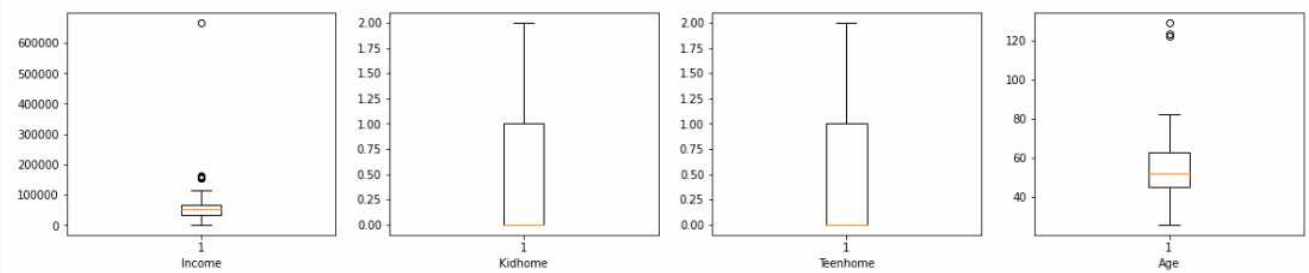
Instance : 64,960 →44,100개

# 03 Preprocessing 데이터 전처리

## Step3. 이상치 제거

* 제품 수요량에 관한 데이터는 사람들의 선호도에 따른 값이라 이상치라고 볼 수 없어서 제외 , 0과 1만 값으로 가지는 컬럼 제외

```
coln = ['Income','Kidhome','Teenhome','Age']
j=0
fig = plt.figure(figsize = (20, 30))
for i in coln:
    plt.subplot(7,4,j+1)
    plt.boxplot(data[i])
    j=j+1
    plt.xlabel(i)
plt.show()
```




연속형 자료들에 대해 이상치가 존재하는지 Box-Plot으로 시각화하여 Income(연간 가구소득), Age(나이) 컬럼에서 이상치가 존재하는 것을 확인

Income이 120,000 이상의 데이터와 Age가 100세 이상인 데이터 삭제

```
# Income과 Age에서 이상치 발견 ---> 소득수준 120000 이상, 나이 100세 이상 제거
data = data.drop(data[(data['Income']>120000)|(data['Age']>100)].index)
```

```
data['Income'].value_counts().sort_index(ascending = False).head()
```

```
113734.0    1
105471.0    1
102692.0    1
102160.0    1
101970.0    1
Name: Income, dtype: int64
```

*Income의 값들을 내림차순 정렬하여 확인한 결과
120,000이상은 제거 된 것 확인

```
data['Age'].value_counts().sort_index(ascending = False).head()
```

```
82    1
81    1
79    6
78    7
77    8
Name: Age, dtype: int64
```

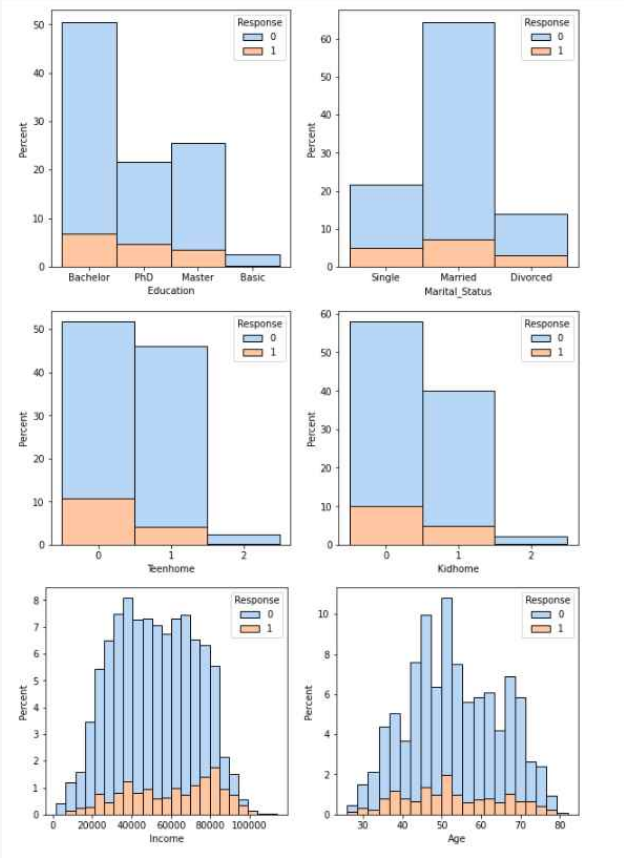
*Age의 값들을 내림차순 정렬하여 확인한 결과
100세 이상은 제거 된 것 확인

# 04 EDA 탐색적 데이터 분석

## Step1. 고객 성향 분석

*고객의 신상정보와 지난 2년간 이벤트 참여 여부를 히스토그램으로 시각화

```
sns.set_palette("pastel")

# Education & Response
plt.figure(figsize=(10,5))
plt.subplot(121)
sns.histplot(data=data, x="Education", hue="Response", multiple="stack", stat="percent")

# Marital_Status & Response
plt.subplot(122)
sns.histplot(data=data, x="Marital_Status", hue="Response", multiple="stack", stat="percent")


# Teen Home & Response
plt.figure(figsize=(10,10))
plt.subplot(221)
sns.histplot(data=data, x="Teenhome", hue="Response", multiple="stack", stat="percent",discrete=True)
plt.xticks([0, 1, 2])


# Kid Home & Response

plt.subplot(222)
sns.histplot(data=data, x="Kidhome", hue="Response", multiple="stack", stat="percent", discrete=True)
plt.xticks([0, 1, 2])


# Income & Response
plt.figure(figsize=(10,12,15))
plt.subplot(321)
sns.histplot(data=data, x="Income", hue="Response", multiple="stack", stat="percent")

# Age & Response
plt.subplot(322)
sns.histplot(data=data, x="Age", hue="Response", multiple="stack", stat="percent")
plt.show()
```




### 이벤트 참여도

1. Education & Response
   **학사** 〉 박사 〉 석사 〉 고졸

2. Marital_Status & Response
   **기혼** 〉 미혼 〉 이혼

3. Teen Home & Response
   **0명** 〉 1명 〉 2명

4. Kid Home & Response
   **0명** 〉 1명 〉 2명

5. Income & Response
   **700,000 ~ 850,000**에서 참여도 높음

6. Age & Response
   **45~50세** 고객들이 참여도 높음

# 04 EDA 탐색적 데이터 분석

## 이벤트 참여도

1. Education & Response
학사 〉 박사 〉 석사 〉 고졸

2. Marital_Status & Response
기혼 〉 미혼 〉 이혼

3. Teen Home & Response
0명 〉 1명 〉 2명

4. Kid Home & Response
0명 〉 1명 〉 2명

5. Income & Response
700,000 ~ 850,000에서 참여도 높음

6. Age & Response
45~50세 고객들이 참여도 높음

## 목표 고객층 선정

이벤트를 기획할 때, 교육수준과 자녀의 수, 소득 수준을 고려하기에는 힘들기 때문에
결혼여부와 나이대만 고려하여 **40~50대 기혼인 고객들을** 대상으로
이벤트를 기획하는 것이 이벤트 참여도를 높이는데 효과가 있을 것.

# 04 EDA 탐색적 데이터 분석

## Step2. 제품 수요량 분석

*40~50대 기혼 고객들의 제품 수요량을 Bar Chart로 시각화

```
# 40~50대 기혼이 아닌 데이터는 삭제하여 data1으로 생성
data1 = data.drop(data[(data['Marital_Status']!='Married')|(data['Age']<40)|(data['Age']>50)].index)
```

```
data1['Marital_Status'].value_counts()
```

```
Married    476
Name: Marital_Status, dtype: int64
```

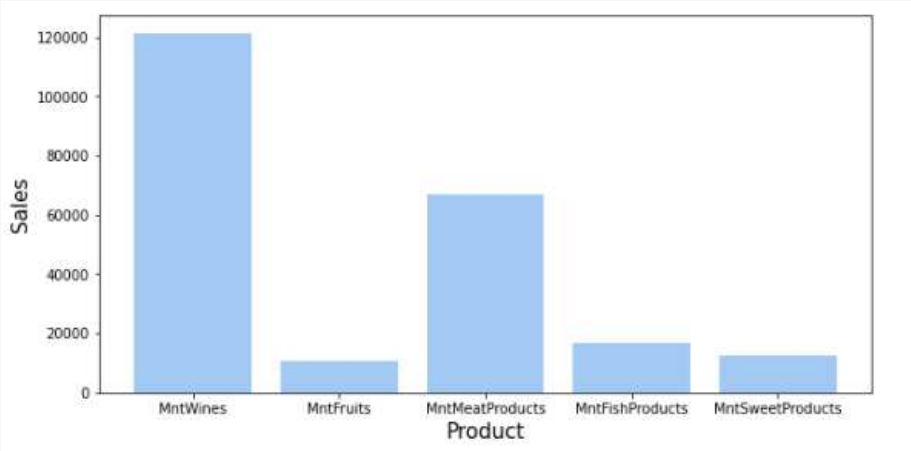
```
data1['Age'].value_counts()
```

```
46    59
50    56
44    54
47    52
48    51
49    47
45    36
43    34
40    33
42    27
41    27
Name: Age, dtype: int64
```

```
sales = data1[['MntWines','MntFruits','MntMeatProducts','MntFishProducts','MntSweetProducts']].sum()
sales.sort_values(ascending = False)
```

```
MntWines            121346
MntMeatProducts      66874
MntFishProducts      16793
MntSweetProducts     12451
MntFruits            10716
dtype: int64
```

```
plt.figure(figsize=(10,5))
x = sales.index
y = sales.values
plt.bar(x,y)
plt.xlabel('Product', fontsize=15)
plt.ylabel('Sales', fontsize=15)
```




와인(121,346건) 〉육류(66,874건) 〉생선(16,793건) 〉간식(12,451건) 〉과일(10,716건)

# 04 EDA 탐색적 데이터 분석

## 40~50대 기혼 고객들의 제품 수요량

**와인**(121,346건) 〉 **육류**(66,874건) 〉 **생선**(16,793건) 〉 **간식**(12,451건) 〉 **과일**(10,716건)

## 목표 제품군 선정

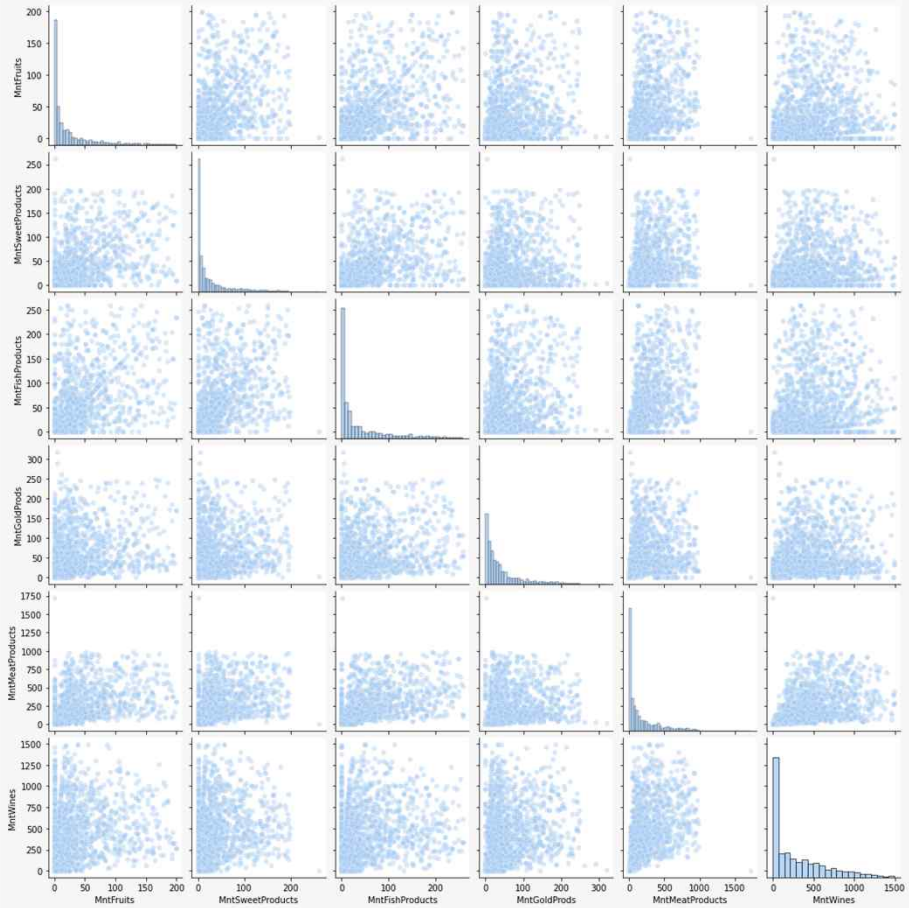
이벤트 참여도가 높은 40~50대 기혼 고객들의 제품 수요량을 분석한 결과

**와인과 육류** 위주로 이벤트를 기획하면 효과가 있을 것

# 04 EDA 탐색적 데이터 분석

## Step3. 제품 간 상관관계 분석

*제품 수요량 간의 상관관계 파악을 위해 산점도로 시각화




```
NUMERICAL_FEATURES = ['MntFruits','MntSweetProducts', 'MntFishProducts', 'MntGoldProds','MntMeatProducts', 'MntWines']

sns.pairplot(data=data[NUMERICAL_FEATURES],
             kind='scatter', plot_kws={'alpha':0.4})
plt.show()
```

### 묶음상품 선정

산점도를 통해 확인한 결과, 상품 간의 상관 관계는 없다고 볼 수 있음

묶음상품(Bundle)로 이벤트를 기획해도 **효과가 없을 것**으로 보임

# 05 Prediction Model 분류예측모델

## Step1. 범주형 자료 → 수치형 자료

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2205 entries, 0 to 2239
Data columns (total 20 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Education            2205 non-null   object
 1   Marital_Status       2205 non-null   object
 2   Income               2205 non-null   float64
 3   Kidhome              2205 non-null   int64
 4   Teenhome             2205 non-null   int64
 5   Recency              2205 non-null   int64
 6   MntWines             2205 non-null   int64
 7   MntFruits            2205 non-null   int64
 8   MntMeatProducts      2205 non-null   int64
 9   MntFishProducts      2205 non-null   int64
 10  MntSweetProducts     2205 non-null   int64
 11  MntGoldProds         2205 non-null   int64
 12  NumDealsPurchases    2205 non-null   int64
 13  NumWebPurchases      2205 non-null   int64
 14  NumCatalogPurchases  2205 non-null   int64
 15  NumStorePurchases    2205 non-null   int64
 16  NumWebVisitsMonth    2205 non-null   int64
 17  Complain             2205 non-null   int64
 18  Response             2205 non-null   int64
 19  Age                  2205 non-null   int64
dtypes: float64(1), int64(17), object(2)
memory usage: 441.8+ KB
```

```
# 교육수준의 경우 레벨에 따라 0,1,2,3으로 변경
data['Education'] = data['Education'].replace(['Basic'],0)
data['Education'] = data['Education'].replace(['Bachelor'],1)
data['Education'] = data['Education'].replace(['Master'],2)
data['Education'] = data['Education'].replace(['PhD'],3)
data['Education'].value_counts()
```

```
1    1113
2     562
3     476
0      54
Name: Education, dtype: int64
```

데이터의 정보에서 'Education' 과 'Marital_Status' 가 object타입인 것을 확인

분류예측모델에서는 범주형 자료는 학습시키지 못하므로 수치형으로 변환 해줘야 함

교육수준 : 고졸 = 0 , 학사 = 1, 석사 = 2, 박사 = 3으로 변경

# 05 Prediction Model 분류예측모델

## Step1. 범주형 자료 → 수치형 자료

```
# 결혼여부는 pandas의 get_dummies()를 이용해 간단하게 원-핫 인코딩 진행.
data = pd.get_dummies(data)
data.head()
```

| NumWebVisitsMonth | Complain | Response | Age | Marital_Status_Divorced | Marital_Status_Married | Marital_Status_Single |
|---|---|---|---|---|---|---|
| 7 | 0 | 1 | 65 | 0 | 0 | 1 |
| 5 | 0 | 0 | 68 | 0 | 0 | 1 |
| 4 | 0 | 0 | 57 | 0 | 1 | 0 |
| 6 | 0 | 0 | 38 | 0 | 1 | 0 |
| 5 | 0 | 0 | 41 | 0 | 1 | 0 |

결혼여부는 원-핫 인코딩 진행하여 기혼/미혼/이혼을 컬럼으로
해당 카테고리에 해당하면 1, 그렇지 않으면 0으로 값을 변환

### * 원-핫 인코딩(One-Hot Encoding)

원-핫 인코딩은 단어 집합의 크기를 벡터의 차원으로 하고, 표현하고 싶은 단어의 인덱스에 1의 값을 부여하고, 다른 인덱스에는 0을 부여하는 단어의 벡터 표현 방식

## 결과

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2205 entries, 0 to 2239
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Education               2205 non-null   int64
 1   Income                  2205 non-null   float64
 2   Kidhome                 2205 non-null   int64
 3   Teenhome                2205 non-null   int64
 4   Recency                 2205 non-null   int64
 5   MntWines                2205 non-null   int64
 6   MntFruits               2205 non-null   int64
 7   MntMeatProducts         2205 non-null   int64
 8   MntFishProducts         2205 non-null   int64
 9   MntSweetProducts        2205 non-null   int64
 10  MntGoldProds            2205 non-null   int64
 11  NumDealsPurchases       2205 non-null   int64
 12  NumWebPurchases         2205 non-null   int64
 13  NumCatalogPurchases     2205 non-null   int64
 14  NumStorePurchases       2205 non-null   int64
 15  NumWebVisitsMonth       2205 non-null   int64
 16  Complain                2205 non-null   int64
 17  Response                2205 non-null   int64
 18  Age                     2205 non-null   int64
 19  Marital_Status_Divorced 2205 non-null   uint8
 20  Marital_Status_Married  2205 non-null   uint8
 21  Marital_Status_Single   2205 non-null   uint8
dtypes: float64(1), int64(18), uint8(3)
memory usage: 431.0 KB
```

모든 컬럼이 수치형 자료로 바뀐 것을 확인 할 수 있음

# 05 Prediction Model 분류예측모델

## Step2. 학습:검정 데이터 분할

```
# 각 변수에 대한 Response를 예측 하는 모델이기에 종속변수를 Response로 설정하여 y 데이터로 저장
x_data = data.drop('Response',axis = 1)
y_data = data['Response']

# 데이터셋 전부를 학습 시키면 분류예측모델의 정확도를 판단 할 수 없기에
# 데이터셋을 학습:검증(7:3) 데이터로 나누는 과정

x_train,x_test,y_train,y_test = train_test_split(x_data, y_data, test_size=0.3, random_state=777, stratify=y_data)
x_train, y_train =SMOTE(random_state=2).fit_resample(x_train, y_train.ravel())
```

```
print('x_train : ',len(x_train))
print('y_train : ',len(y_train))
print('x_test : ',len(x_test))
print('y_test : ',len(y_test))
```

```
x_train :  2620
y_train :  2620
x_test :  662
y_test :  662
```

[독립변수 = Response, 종속변수 = Response 를 제외한 컬럼] 으로 x_data와 y_data로 분할

학습 : 검증 데이터는 train_test_split을 이용해 7:3으로 분할

x_train (독립변수 학습데이터) : 2620개

y_train (종속변수 학습데이터) : 2620개

x_test (독립변수 검증데이터) : 662개

y_test (종속변수 검증데이터) : 662개

# 05 Prediction Model
# 분류예측모델

## Step3. 예측모델 생성 (SVM vs RandomForest)

```
#SVM(기본)
model_svm = svm.SVC().fit(x_train, y_train)
predict_svm = model_svm.predict(x_test)
print('SVM :',accuracy_score(y_test,predict_svm))

#RandomForest(기본)
model_rf = RandomForestClassifier()
model_rf.fit(x_train,y_train)
predict_rf = model_rf.predict(x_test)
print('RandomForest :',accuracy_score(y_test,predict_rf))
```

```
SVM : 0.7492447129909365
RandomForest : 0.8685800604229608
```

분류예측 모델로 서포트벡터머신(SVM) 과 랜덤포레스트를 사용

두 가지 모델 모두 기본 설정으로 실행한 결과

SVM의 정확도 : 0.749

RandomForest 의 정확도 : 0.869

수치만 보면 RandomForest를 선정하는 것이 옳으나 SVM같은 경우 데이터 스케일링, 하이퍼 파라미터 튜닝에 따라 정확도의 편차가 심하기때문에 두 가지 모델 전부 진행해보고 비교하기로 함

# 05 Prediction Model
# 분류예측모델

## Step4. 데이터 스케일링

```
# 데이터 스케일링 진행(StandardScaler사용)
scaler = StandardScaler()
scaler.fit(x_data)
x_data = scaler.transform(x_data)

x_train,x_test,y_train,y_test = train_test_split(x_data, y_data, test_size=0.3, random_state=777, stratify=y_data)
x_train, y_train =SMOTE(random_state=2).fit_resample(x_train, y_train.ravel())
```

```
#SVM(스케일링)
model_svm = svm.SVC().fit(x_train, y_train)
predict_svm = model_svm.predict(x_test)
print('SVM :',round(accuracy_score(y_test,predict_svm),3))

#RandomForest(스케일링)
model_rf = RandomForestClassifier()
model_rf.fit(x_train,y_train)
predict_rf = model_rf.predict(x_test)
print('RandomForest :',round(accuracy_score(y_test,predict_rf),3))
```

```
SVM : 0.811
RandomForest : 0.878
```

sklearn의 StandardScaler모듈을 이용하여

x_data를 평균 = 0 , 표준편차 = 1이 되도록 표준화 진행한 후,

SVM 과 RandomForest 두 가지 모델의 정확도를 비교

SVM : 0.811

RandomForest : 0.878

데이터 스케일링을 진행한 후에도 RandomForest의 정확도가 더 높았음

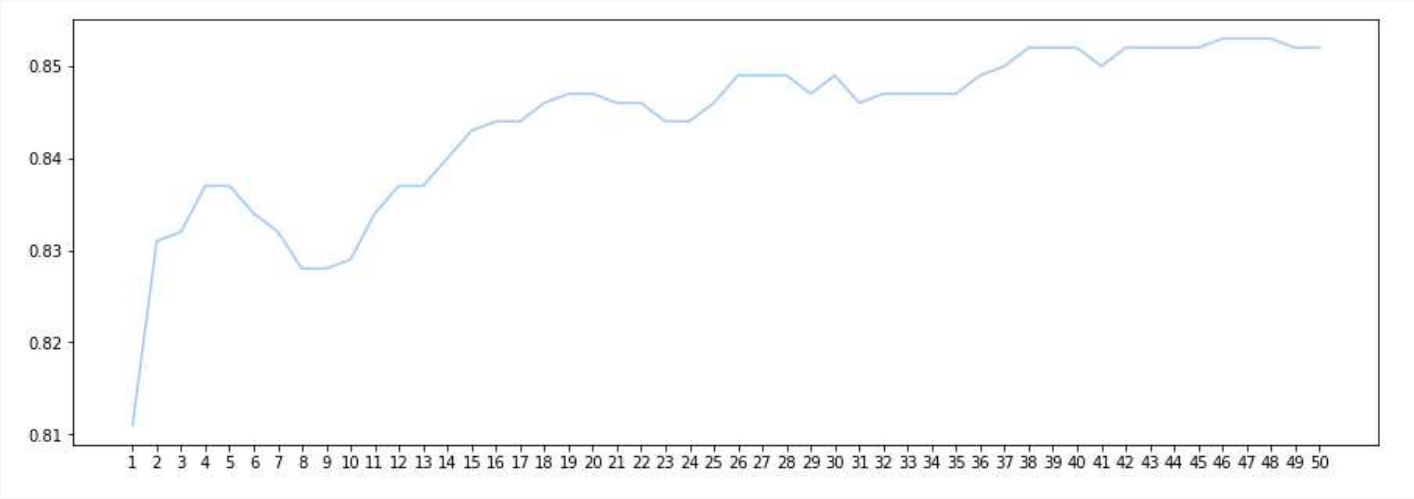
# 05 Prediction Model 분류예측모델

## Step5. 하이퍼 파라미터 튜닝 - SVM

```
#하이퍼 파라미터 수정(SVM - C)
# 가장 높게 나온 C=48으로 선정
score_list = []
x_list = []
for i in range(1,51) :
  model_svm = svm.SVC(C=i).fit(x_train, y_train)
  predict_svm = model_svm.predict(x_test)
  score_list.append(round(accuracy_score(y_test,predict_svm),3))
  x_list.append(i)

plt.subplots(figsize=(15, 5))
plt.plot(x_list, score_list)
plt.xticks(x_list)
plt.show()
```

SVM의 C값을 for문을 이용해 1~50까지 올려가면서 그래프를 그려본 결과 C=48일 때 가장 높은 정확도를 보여줬음

SVM의 하이퍼파라미터로 gamma 설정도 있으나, gamma 수정은 정확도가 미미하게 바뀌어 제외하였음



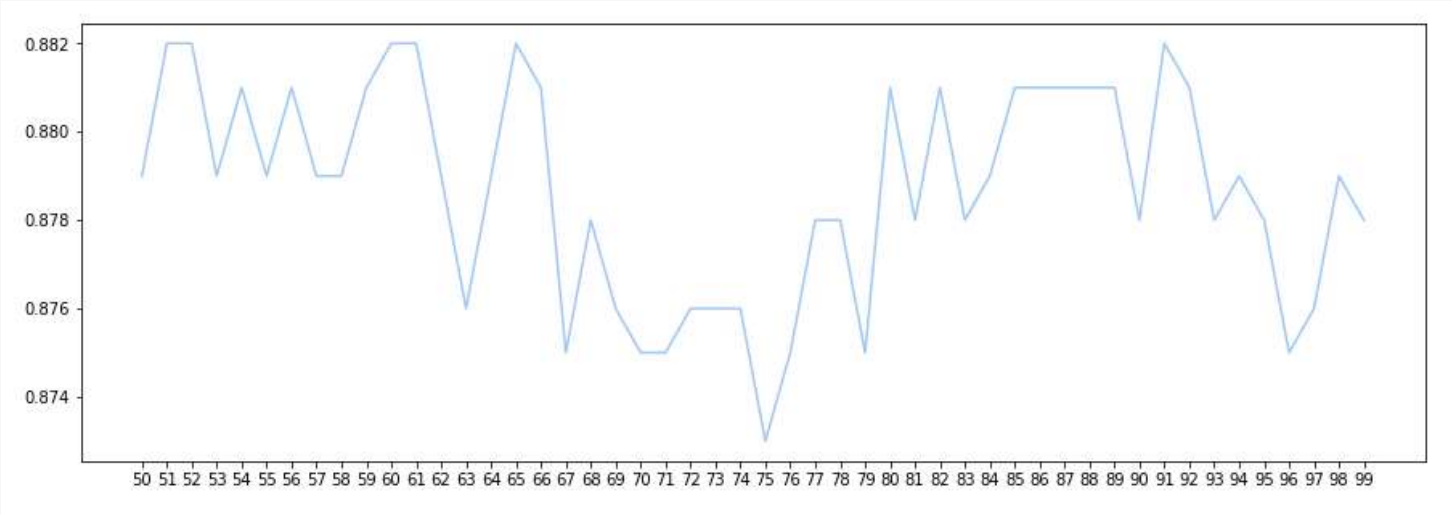
# 05 Prediction Model 분류예측모델

## Step5. 하이퍼 파라미터 튜닝 - RandomForest

```
#RandomForest(생성할 Tree갯수)
# tree  = 59개 일 때 최대수치
score_list = []
x_list = []
for i in range(50,100) :
  model_rf = RandomForestClassifier(n_estimators=i, max_depth=100,random_state=0)
  model_rf.fit(x_train,y_train)
  predict_rf = model_rf.predict(x_test)
  score_list.append(round(accuracy_score(y_test,predict_rf),3))
  x_list.append(i)

plt.subplots(figsize=(15, 5))
plt.plot(x_list, score_list)
plt.xticks(x_list)
plt.show()
```

RandomForest의 n_estimators(Tree의 개수)를 50~99까지 for문을 이용해 정확도를 시각화한 **결과 Tree가 59**개일 때, 가장 높은 정확도를 보여줬음

Tree의 깊이를 설정하는 max_depth의 경우에도 100 전후로 미미한 수치 변화만 보여주어 max_depth = 100으로 설정



# 05 Prediction Model 분류예측모델

## Step6. 예측모델 평가 및 선정

```
#예측 모델 비교

#SVM
model_svm = svm.SVC(C=48).fit(x_train, y_train)
predict_svm = model_svm.predict(x_test)
print('SVM :',round(accuracy_score(y_test,predict_svm),3))


#RandomForest
model_rf = RandomForestClassifier(n_estimators=59, max_depth=100,random_state=0)
model_rf.fit(x_train,y_train)
predict_rf = model_rf.predict(x_test)
print('RandomForest :',round(accuracy_score(y_test,predict_rf),3))
```

```
SVM : 0.853
RandomForest : 0.881
```

데이터 스케일링, 하이퍼 파라미터를 튜닝한 두 가지 모델의 정확도를 비교한 결과 RandomForest를 선정

SVM : 0.853

RandomForest : 0.881

이는 고객데이터를 test로 넣었을 때, **88.1%의 정확도로 고객의 이벤트 참여여부를 예측**할 수 있다는 의미

# 06 Conclusion 결론

- **40~50대 기혼고객**에게 와인과 육류 위주의 이벤트를 기획한다면 이벤트 효과를 볼 수 있을 것으로 보임
- 제품 수요 간 상관관계가 없으므로 **Bundle의 효과는 없을 것**으로 보임
- 새로운 이벤트를 기획한 후, 특정 고객들이 이벤트를 참여할 것인지에 대한 여부는 예측모델을 통해 목표 고객들의 Test데이터만 넣어준다면 **88.1%의 정확도로 고객의 이벤트 참여여부를 예측**할 수 있음

| 고객 성향 분석 | 제품 수요량 분석 | 제품 간 상관관계 분석 | 예측모델 개발 |
|---|---|---|---|
| 40~50대 기혼 고객 | 와인, 육류 | 제품 수요 간 상관관계 없음 | 88.1%의 정확도로<br>고객반응 예측 가능 |


END