고급 통계적 머신러닝 프로젝트 기말 레포트

2조(신나리, 이준희, 전보민, 정준식)

I. Introduction

기업에서 마케팅의 효율을 높이기 위해서 필요한 것은 고객의 특성을 잘 파악하는 것이다. 따라서 본 레포트에서는 EDA를 통해 고객 관련 변수들의 특성을 확인하고, 클러스터링을 통해 고객군을 분류하였다. 또한 어떤 변수들이 고객의 소비 및 마케팅 캠페인에 대한 참여에 영향을 주는지 알아보기 위해 회귀와 분류를 진행하였다. 분석을 위한 데이터로는 Kaggle의 'Marketing Campaign' 데이터를 사용하였다. 이후의 내용은 데이터 설명, EDA, 클러스터링, 회귀, 분류 순으로 진행된 데이터 분석 결과, 그리고 결론 순으로 기술하였다.

II. Explanation of Data

'Marketing Campaign' 데이터는 총 2240개의 관측치와 27개의 변수로 구성되어 있으며, 변수의 종류는 아래와 같다.

- ID, Year_Birth, Education, Marital_Status, Income, Kidhome, Teenhome, Dt_Customer, Recency, Complain
- MntWines, MntFruits, MntMeatProducts, MntFishProducts, MntSweetProducts, MntGoldProducts
- NumDealsPurchase, AcceptedCmp1, AcceptedCmp2, AcceptedCmp3, AcceptedCmp4, AcceptedCmp5, Response
- NumWebPurchases, NumCatalogPurchases, NumStorePurchases, NumWebVisitsMonth

ID는 고객의 ID를, Year_Birth는 태어난 년도를 나타낸다. Education은 범주형 변수로, 'Basic', '2n Cycle', 'Graduation', 'Master', 'PhD'의 다섯 가지 카테고리로 구성되어 있으며 Marital_Status는 'Absurd', 'Alone', 'Divorced', 'Married', 'Single', 'Together', 'Widow', 'YOLO'의 여덟 가지 카테고리로 구성되어 있다. Income은 고객의 소득을 나타내며 단위는 달러이고, Kidhome과 Teenhome은 각각 가정 내 아이의 수와 청소년의 수를 의미한다. Dt_Customer은 고객이 가입한 날짜를 나타내며, Recency는 고객이 마지막으로 물건을 구입한 날짜로부터 지난 날짜를 나타낸다. Complain은 고객의 컴플레인 경험 여부를 나타내고, MntWines, MntFruits, MntMeatProducts, MntFishProducts, MntSweetProducts, MntGoldProducts는 각각 와인, 과일, 축산물, 수산물, 과자, 그리고 금에 대한 고객의 소비금액을 나타내며, 단위는 달러이다. NumDealsPurchase는 고객이 할인된 가격으로 물건을 구매한 횟수를 나타내며, AcceptedCmp1, AcceptedCmp2, AcceptedCmp3, AcceptedCmp4, AcceptedCmp5, Response는 각각 고객이 첫 번째부터 마지막(여섯 번째) 마케팅 캠페인에 참여한 여부를 나타내는 더미 변수이다. NumWebPurchases, NumCatalogPurchases, NumStorePurchases는 각각 고객이 온라인, 카탈로그, 오프라인으로 물건을 구매한 횟수를 의미하는 변수이며, NumWebVisitsMonths는 고객이 한달 간 웹사이트에 방문한 횟수를 나타낸다.

분석에 앞서 데이터 전처리과정을 통해 이상치를 제거하고 분석에 필요한 새로운 변수를 정의한

정제된 데이터를 완성하였다. Income 변수에 존재하는 24개의 N/A 값은 전체 데이터의 크기인 2240개에 비해 적다고 판단하여 제거하였으며, Income 변수에 666666이라는 지나치게 큰 이상치가 존재하여 제거하였다. 또한 (2021 - Year_Birth)를 'Age'라는 새로운 변수로 정의하였고, 여기에서 발견되는 지나치게 큰 3개의 값 또한 제거하였다. Marital_Status의 경우 YOLO, Absurd 그룹의 인원이 3명이하로 매우 적어 이에 해당하는 고객의 데이터를 제거하고, 분석의 용이성을 위해 새롭게 (Divorced, Single, Widow, Alone) 범주를 묶어 'Alone'으로 정의하고, (Married, Together) 범주를 묶어 'Together'로 정의하였다. Education의 경우도 마찬가지로 (Basic, 2n Cycle) 범주를 묶어 'High School'로, (Master, PhD) 범주를 묶어 'PostGraduate'로, 기존의 Graduation 범주를 'Graduate'로 정의하였다. Dt_Customer 변수 또한 기존의 날짜형태의 변수에서 Q1, Q3 값을 기준으로 하여 'Old', 'Middle', 'New'의 세 가지 카테고리로 분류하였다. 이후 고객의 식비 회귀 모형 적합을 위해 (MntWines + MntMntWines + MntFruits + MntMeatProducts + MntFishProducts + MntSweetProducts)를 'Food Expense'라는 변수로 정의하였으며, 'NumChild'는 Kidhome과 Teenhome 변수의 합으로, 'NumResponded'는 캠페인에 응답한 횟수의 총합으로 정의하였다.

III. Analysis & Results

1) EDA

먼저, 변수들의 특성을 파악하기 위해 EDA를 진행하였다. Appendix에 첨부된 그림 5 – 그림 15는 EDA를 위해 활용된 플롯을 나타낸다. 지면 상의 문제로 본 레포트에서는 중요한 그림만을 다루기로 하며, 자세한 설명은 Appendix의 그림 설명을 참조하길 바란다. 먼저, 그림 5의 Income Boxplot에서 중앙값은 5만 달러 정도로 나타나며 IQR의 값도 35000정도로 그렇게 크지 않은 데에 반해 소득이 15만 달러를 넘는 5개의 매우 큰 값들이 존재하는 것을 확인할 수 있다. 또한, 그림 10을 통해 소비 그래프가 오른쪽으로 꼬리가 긴 형태를 나타내는 것을 확인할 수 있다. 이는 대부분의 고객은 소비 금액이 적지만, 일부 고객이 많은 돈을 소비하고 있음을 나타낸다. 그림 12를 통해 캠페인에 응답한 고객의 수가 전체의 약 5%에 밖에 달하지 않음을 확인할 수 있으며, 그림 13은 카테고리 별 소비량의 violin plot을 나타낸다. 이를 통해 Fish, Fruit, Gold, Sweet는 전체적으로 편차도 적으며 대부분의 고객의 소비금액이 적지만, Meat와 Wine의 경우 고객 별 소비 금액의 편차가 크며 꼬리가 긴 분포를 띄는 것을 확인할 수 있다. 이는 Meat와 Wine의 경우 상대적으로 가격이 높은 식품에 속하기 때문으로 추측된다. 그림 15는 변수들 간의 상관관계를 나타낸 그래프이며, 이를 통해 더미 변수를 사용하여 나타낸 변수들을 제외하고는 변수들 간 큰 상관관계가 존재하지 않음을 확인할 수 있다.

2) 클러스터링 (Clustering)

다음으로는 클러스터링을 진행하기 위한 추가적인 전처리를 진행했다. 우선, 자료들이 수치형으로 인식되어야 하기에 범주형 변수들을 라벨 인코딩하였다. 또한, 상관관계가 높은 변수들을 직교화 시키고, 클러스터링 속도 개선 및 군집화 정도 개선을 위해 PCA를 진행했다. EDA에서 skewed한 변수들이 있었고, 변수들의 척도가 다른 점을 확인할 수 있었는데,

skewed한 column을 정규분포화 시키고, scale-variant한 PCA의 특성을 고려해 PCA 전에 standardscaler를 사용해 주었다. Component의 개수는 추가 분산 설명량이 0.1 이하가 되는 3개를 사용했다. 그림 16과 17을 통해 해당 과정을 확인할 수 있다.

Cluster의 수를 결정하기 위한 기준으로는 distortion score를 사용했다. 이 때, distortion score는 각각의 점이 배정된 클러스터의 중심으로부터 떨어진 거리의 제곱합의 평균으로 계산되고, K-means clustering의 distortion score elbow point를 기준으로 클러스터의 수를 결정했다. 그림 18을 통해 elbow point가 4인 것을 확인할 수 있다. 이후, cluster의 수를 4개로 계층적 클러스터링을 진행했다. K-means로 군집의 수를 결정한 이후에 계층적 클러스터링을 진행한 이유는 ward linkage를 사용해서 비교적 크기가 비슷한 군집을 형성하고자 했기 때문이다. 이 때, Ward linkage는 모든 클러스터 내 분산을 가장 작게 증가시키는 두 클러스터를 합치는 방향으로 클러스터링을 진행하고 크기가 유사한 군집을 생성해주는 방법이다.

그림 19의 PCA 3D plot을 통해 군집이 잘 분리되었음을 확인할 수 있다. 그림 20에서 군집의 크기 또한 비슷하게 형성이 되었고, 그림 21에서 Money와 Income변수를 축으로 하는 산점도를 통해 Money와 Income에 따라 클러스터가 유의미한 차이를 보임을 알 수 있다. 그림 22를 통해 모든 클러스터에서 2번 이상 캠페인에 호응을 한 사람들의 수는 적은 것을 알 수 있고, 그림 23에서 0번 클러스터에 해당하는 사람들이 할인을 많이 받았음을 알 수 있다. 그림 24에는 클러스터의 특징을 고찰하고자 소비액을 Y축으로 Kidhome, Teenhome, Customer_For, Age, Children, Family_Size, Is_Parent, Education, Living_With 총 9개의 변수들을 X축으로 KDE plot을 그렸다. 이 때, KDE plot을 그릴 때 X축의 범주형 변수들의 경우, 각 범주에 따라 퍼진 정도를 나타내기 위해 라벨인코딩을 통해 연속형으로 표현을 하였다. 그래야만 X축 변수의 특정 범주 level에서 spent가 분포하는 정도를 나타낼 수 있기 때문이다.

해당 분석을 통해 cluster의 특징을 추출한 결과는 다음과 같다. 클러스터 0번의 경우 상대적으로 연령대가 높고, 청소년 자녀를 둔 부모인 경우가 많았다. 클러스터 1번의 경우 부모가 아닌 경우가 많으며 최대 가족 구성원 수가 2명이고 소득이 높은 편으로 나타났다. 클러스터 2번의 경우 상대적으로 연령대가 낮고, 어린 자녀를 둔 부모인 경우가 많았다. 마지막으로 클러스터 3번의 경우 상대적으로 연령대가 높고, 소득과 학력이 낮은 경우가 많았다. 클러스터 특징을 파악을 통해 클러스터 별 특성에 맞는 맞춤형 마케팅을 진행할 수 있을 것으로 기대된다.

3) 회귀 (Regression)

회귀 분석을 통해서는 고객의 특성이 소비에 미치는 영향을 알아보았다. 이때 설명변수로는 Age, Education, Marital_Status, NumChild, Customer_period, NumResponded, Income을 상정했으며, 종속변수로는 foodexpense을 사용하였다. 예측력보다는 설명력이 높은 모델을 만들기 위해 해석이 용이한 Linear Regression, Decision Tree, Random Forest, Generalized Additive Model을 사용했다. 또한 데이터셋을 7대3의 비율로 train set과 test set으로 나누어 train set으로 분석을 진행한 후 최종적으로 test set을 활용해 모델 간 성능을 비교했다.

먼저 Linear Regression을 진행했다. 회귀 계수의 중요성이 왜곡되는 것을 방지하기 위해 수치형 변수들에 대해서 표준화를 진행한 후 분석했다. Stepwise variable selection을 사용해 유의하지 않은 변수들을 제거했으며 최종 회귀 분석 결과는 그림 25에 나타나 있다. 가입한지 오래된 고객일수록, 캠페인에 반응한 횟수가 많을수록, 소득이 많을수록 식비가 높은 경향이 있었으며, 반대로 자식수가 많을수록 음식에 지출한 돈이 적은 경향이 있었다. 일반적으로 자식수가 많을수록 소비도 많아질 것이라고 예상할 수 있는데, 이와 반대되는 결과가 나온 이유는 자식이 있는 부모 고객은 배우자와 소비를 나누어서 하기 때문이라고 해석해볼 수 있었으며, 해당 모델에서 고려하지 못한 interaction effect가 존재하기 때문일 수도 있을 것이다.

다음은 Decision Tree 분석 결과이다. rpart 패키지의 default cp(=0.01)값을 활용해 비교적 단순한 트리를 생성한 후 시각화한 결과가 그림 26에 나타나 있다. 결과를 해석해보면, 소득이 가장 영향력 있는 변수로 나타나 첫번째 분리 규칙으로 사용되었으며 소득이 많을수록, 가입한지 오래된 고객일수록, 캠페인에 반응한 횟수가 많을수록 음식에 지출한 평균 금액이 높은 것으로 나타나 앞서 진행한 Linear Regression 결과와 유사함을 알 수 있다.

그림 27는 Random Forest 분석을 진행한 후 변수 중요도를 시각화한 결과를 보여준다. 총 500개의 트리가 사용되었으며 분리 규칙에 고려된 변수의 개수는 2개이다. 식비에 영향을 미치는 변수의 순서는 소득, 자식 수, 캠페인에 반응한 횟수, 나이 등의 순임을 알 수 있다.

마지막으로, 모델에 flexibility를 부여하여 변수별로 적합한 함수를 찾기 위해 Generalized Additive Model을 활용했다. 연속형 변수인 Income과 Age에 대해 natural spline을 적용했으며, validation set approach를 통해 [2, 5] 사이의 값 중 validation MSE를 최소로 하는 자유도를 결정했다. 그 결과 Income의 자유도는 4로, Age의 자유도는 3으로 설정했다. 모델 적합 결과는 그림 28에 나타나 있다. 소득이 증가하면 식료품 관련 소비가 빠르게 증가하지만 다른 고객들보다 소득이 매우 높은 일부 고객이 존재해 종모양의 형태를 보이고 있다. 나이 변수 또한 단순한 선형 관계가 아니라 40~50대에서 가장 높은 값을 가지는 형태로 나타난다.

최종적으로 모델의 성능을 비교해본 결과, 10-fold cross validation RMSE와 test RMSE 모두 Random Forest가 가장 우수한 성능을 보였다

4) 분류 (Classification)

분류에서는 가장 최근 캠페인에 대한 참여여부를 나타내는 Response 변수를 반응변수로 사용하였다. 이전과 마찬가지로 Train set과 test set의 분할을 7대3으로 진행하였으며 train set의 약 84%가 0에 해당하는 불균형 데이터이기 때문에, scaling 이후 SMOTE(합성 소수 샘플링 기술 알고리즘)을 사용한 오버 샘플링을 진행했다. 그림 29에 해당 변화로 인한 반응변수의 분포가 나타나며 분류 모델은 Logistic Regression, Adaboost, SVM을 사용하였다.

10-fold cross validation 결과 Logistic Regression의 ROC curve AUC는 약 0.8924였으며, 결과는 그림 30과 같다. 고객의 마지막 구매 후 시간이 오래 지날수록, 또한 store에서 구매를 많이 하는 고객의 경우, 캠페인 참여에 대한 오즈비는 감소하였고, Web이나 catalog를 통해

구매를 많이 하는 고객의 경우, 캠페인 참여에 대한 오즈비가 증가한다고 해석할 수 있다.

Adaboost의 경우, learning rate, max_depth, n_estimators에 대해 grid search하였다. 10-fold cross validation 결과 AUC 면적이 가장 큰 값은 0.9826이였으며, 해당 parameter는 max_depth는 15, learning_rate는 0.4, n_estimator는 600이었다.

SVM의 경우, Gaussian Kernel을 사용해서 parameter tuning을 한 결과, cross validation ROC curve AUC는 0.9867이였으며, 해당 parameter는 C는 7.5, gamma는 2이다. 또한 Kernel SVM의 경우, 변수 선택을 하는 것이 어렵다고 알려져 있어, 이전 logistic regression에서 고른 변수들 만을 사용한 결과, ROC curve AUC를 기준으로 성능의 큰 차이를 보이지 않았다.

그림 31을 통해 Cross validation mean AUC의 경우, SVM의 성능이 가장 좋았으나, test set AUC는 Logistic regression의 성능이 가장 좋다는 것을 알 수 있다. 따라서 본 데이터에 대해서, 더 flexible한 모델들이 과적합을 일으키는 것을 확인할 수 있다.

Confusion Matrix에서 Precision은 예측한 것이 실제로 맞은 비율이고, recall은 실제 값에 대해 예측이 바르게 된 비율이다. 따라서, 그림 32의 confusion matrix를 통해 최종 모델인 Logistic Regression에서 실제 캠페인 참여할 고객을 잘 맞추는 것이 중요하다면, 낮은 cut-off value값을 사용하고, 캠페인에 참여할 고객을 예측한 결과에 대한 정확도가 중요하다면, 높은 cut-off value 값을 사용하는 것이 좋을 것이다.

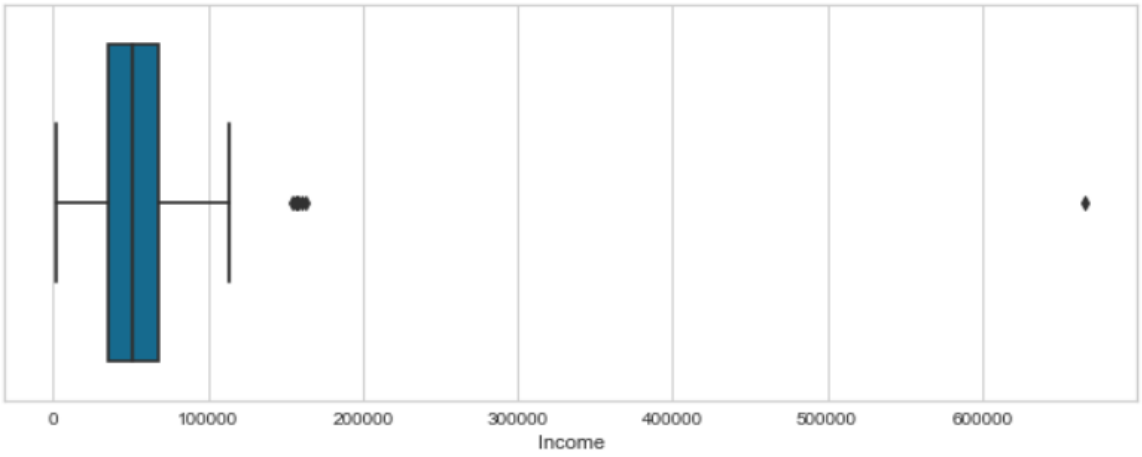
IV. Conclusion & Discussion

지금까지 EDA, 클러스터링, 회귀, 분류 방법을 사용하여 'Marketing Campaign' 데이터에 대한 분석을 진행해보았다. EDA를 통해 변수들의 분포를 확인함으로써 데이터의 구조에 대해서 확인할 수 있었고, 계층적 클러스터링을 사용하여 고객군을 4개의 클러스터로 나누어 각각의 클러스터가 가지는 특징에 대해서도 살펴보았다. 이때, 클러스터를 구분하는 유의한 기준은 소득, 나이, 자녀수, 학력 등이 있었다. 회귀 분석에서는 선형 회귀 모형, 결정 나무 모형, 랜덤 포레스트 모형, GAM 모형을 사용하여 어떠한 변수들이 고객의 식비에 유의한 영향을 미치는지 살펴보기도 하였다. 모형의 성능을 평가하는 지표로 RMSE를 사용하였고, 랜덤 포레스트 모형이 가장 좋은 성능을 내는 것으로 판단되었다. 분류 모형에서는 로지스틱 회귀 모형, Adaboost, SVM을 사용하여 Response 변수, 즉 가장 최근 있었던 캠페인에 대한 참여 여부를 예측하기도 하였다. 분류 모형의 성능을 평가하는 지표로는 AUC를 사용하였으며, test set에 대한 AUC는 로지스틱 회귀 모형에서 가장 높게 나타나는 것을 확인할 수 있었다.

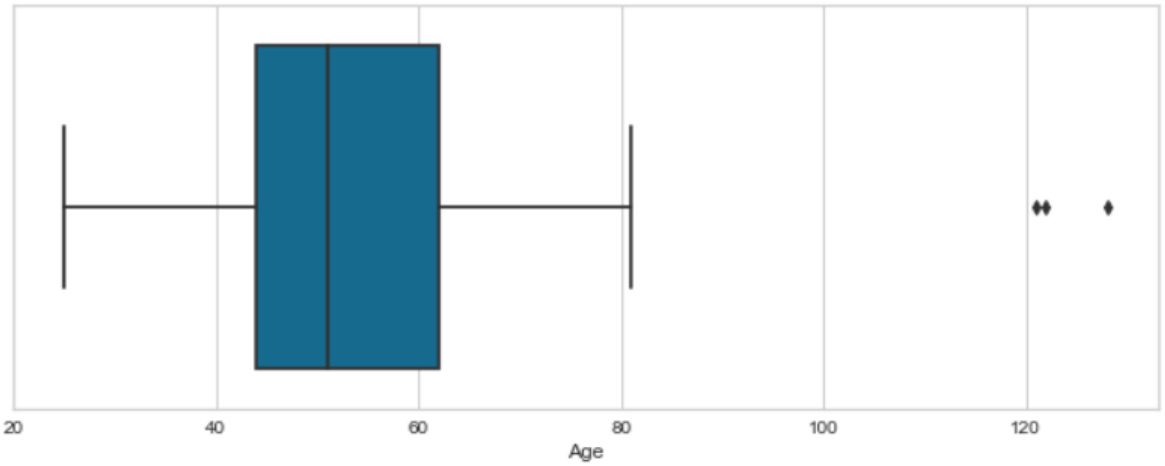
본 레포트에서는 분석을 위주로 진행하였으나 추가적인 적용 방안에 대해 생각한다면 더 흥미로운 연구가 될 것이라 생각한다. 클러스터링의 경우 각 군집에 대한 특성을 파악했으니 어떠한 방식의 마케팅이 어울릴지 생각해보는 것이 의미가 있을 것이다. 이에 관하여 회귀 분석에서 유의하게 나온 변수들을 고려해볼 수 있을 것이다. 또한 로지스틱 회귀 모형을 통해 마케팅 캠페인에 참여하는 사람들의 특성을 고려하여 보다 더 많은 사람들이 캠페인에 참여할 수 있는 방안 또한 생각해볼 수 있을 것이다.

[Appendix]

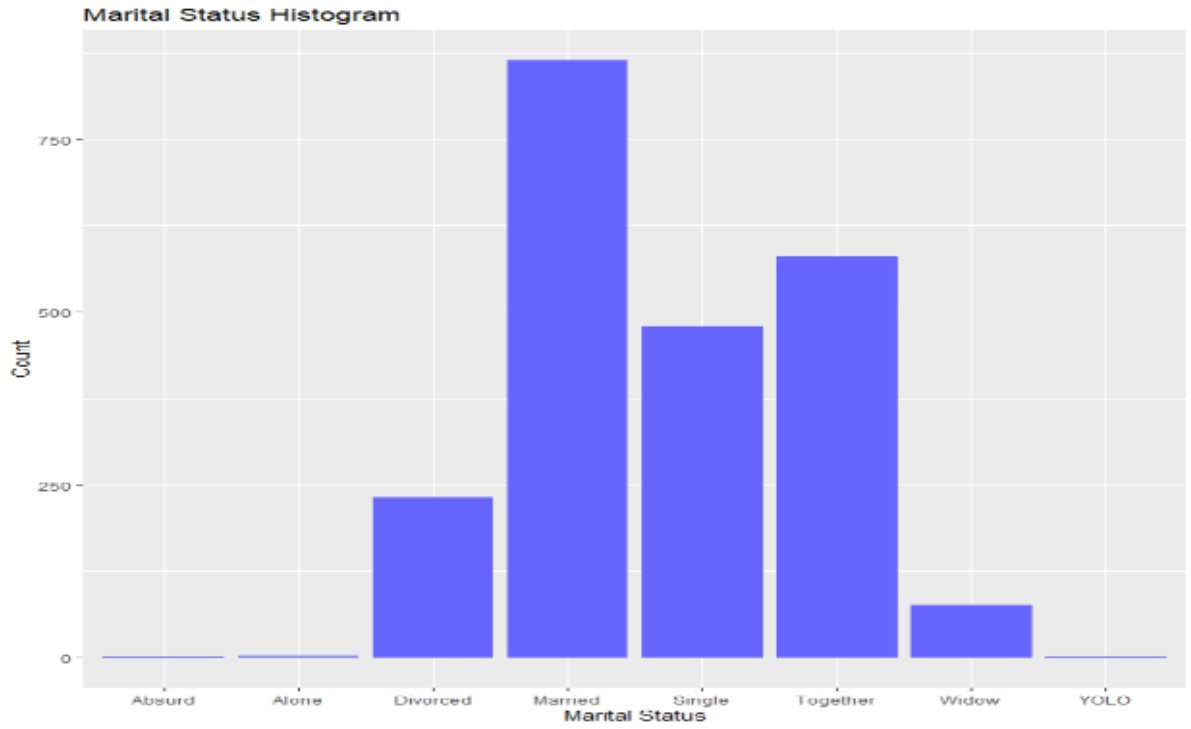
- **Figures**



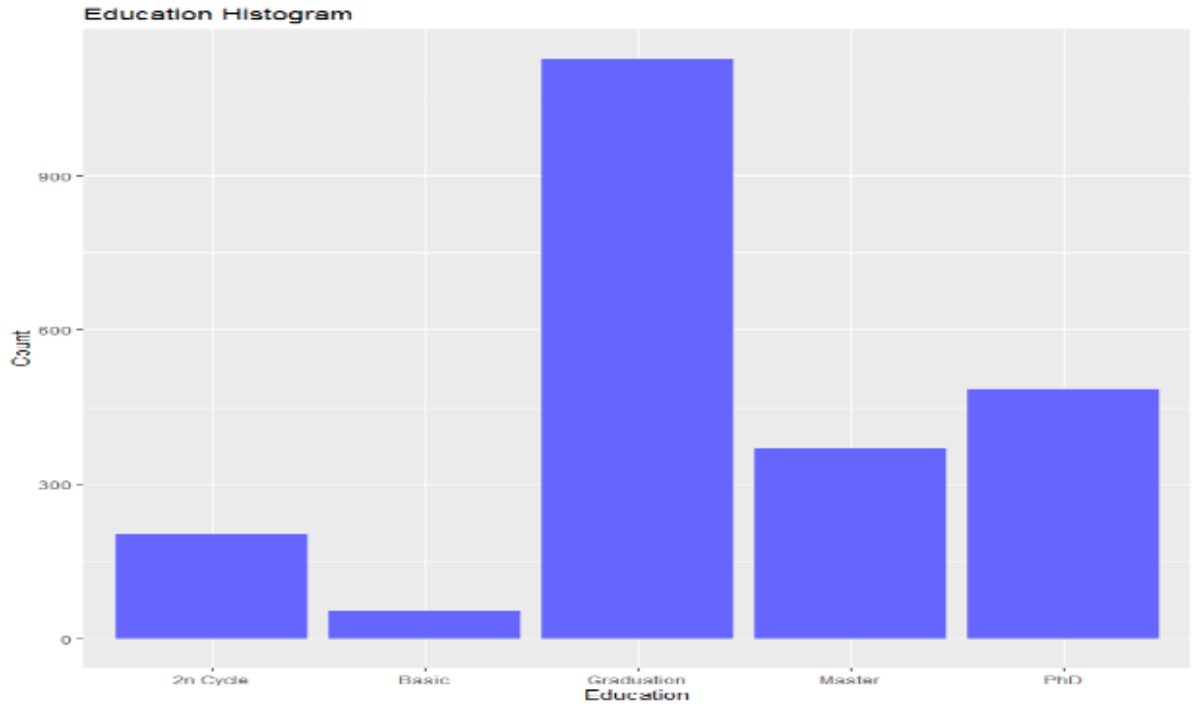
[그림 1. Income Boxplot] 전처리를 하지 않은 데이터의 Income 변수의 boxplot이다. 이를 통해 우측에 한 개의 이상치가 존재하는 것을 확인할 수 있다.



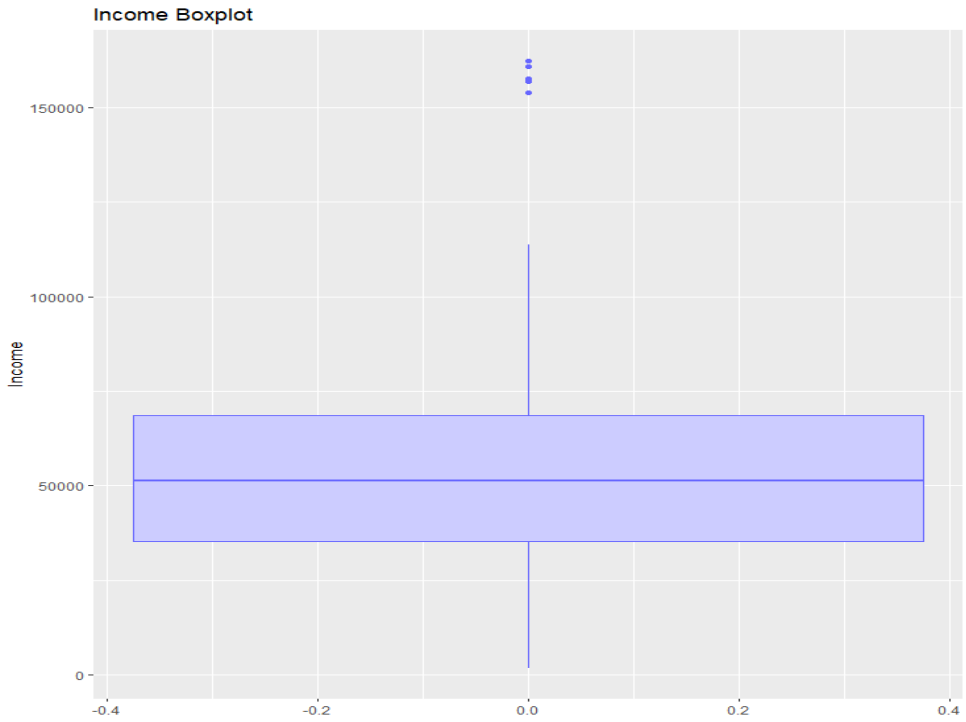
[그림 2. Age Boxplot] 전처리를 하지 않은 데이터의 Age 변수의 boxplot이다. 이를 통해 우측에 세 개의 이상치가 존재하는 것을 확인할 수 있다.



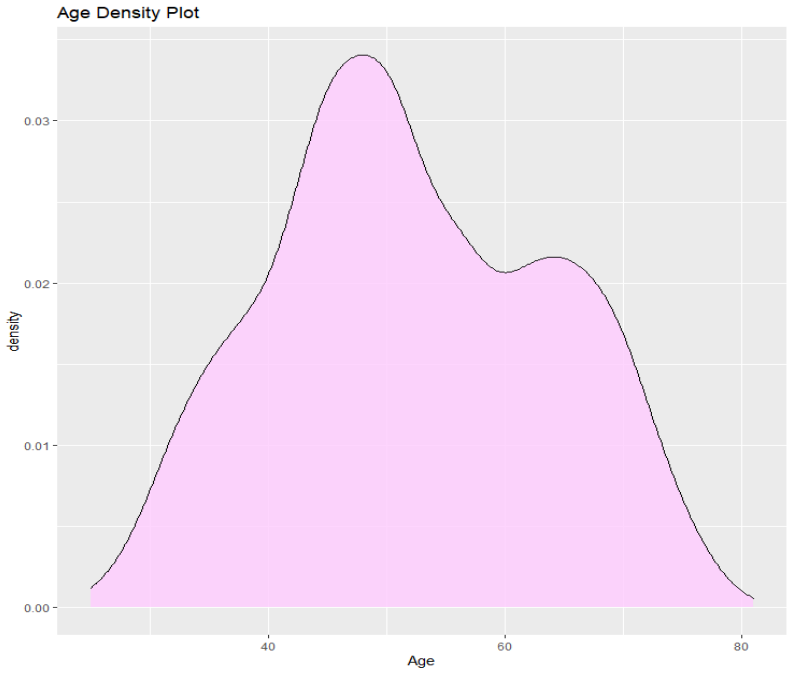
[그림 3. Marital Status Histogram] 전처리를 하지 않은 데이터의 Marital Status 변수의 히스토그램이다. 그래프를 통해 Absurd, YOLO, Alone 범주에 소속된 고객의 수가 매우 적은 것을 확인할 수 있다.



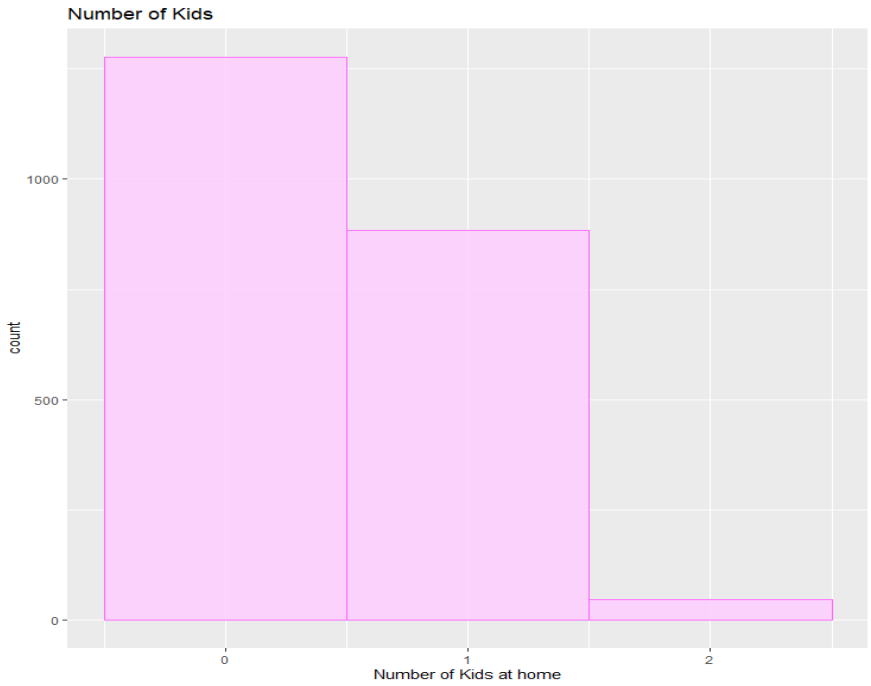
[그림 4. Education Histogram] 전처리를 하지 않은 데이터의 Education 변수의 히스토그램이다. 히스토그램을 통해 Graduation 범주에 속하는 고객의 수가 가장 많고, 2n Cycle과 Basic 범주에 속하는 고객의 수는 상대적으로 적은 것을 확인할 수 있다.



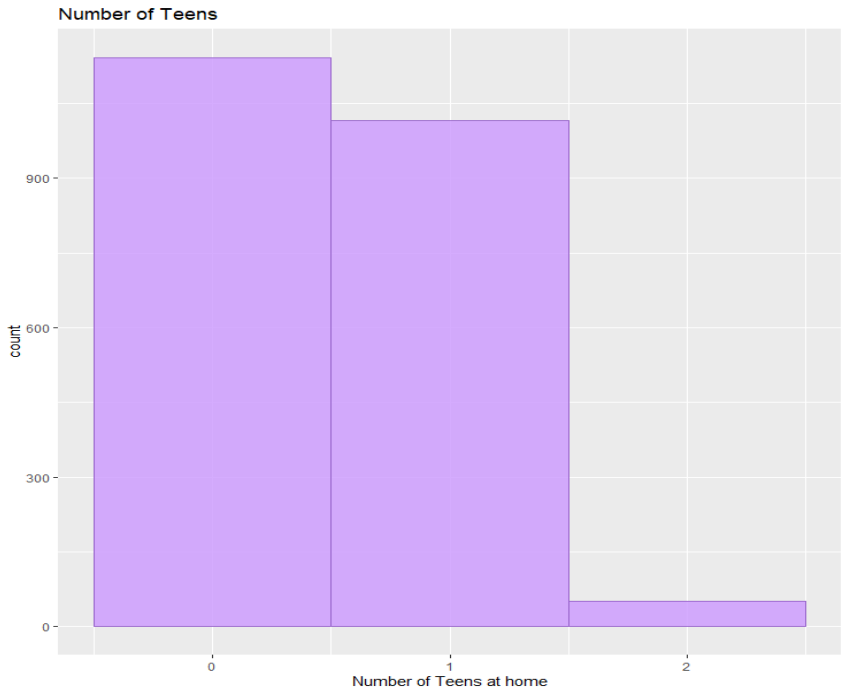
[그림 5. Income Boxplot] 전처리를 한 데이터의 Income 변수의 boxplot이다. 전처리를 한 이후에도 상대적으로 값이 큰 5개의 점이 존재하는 것을 확인할 수 있다.



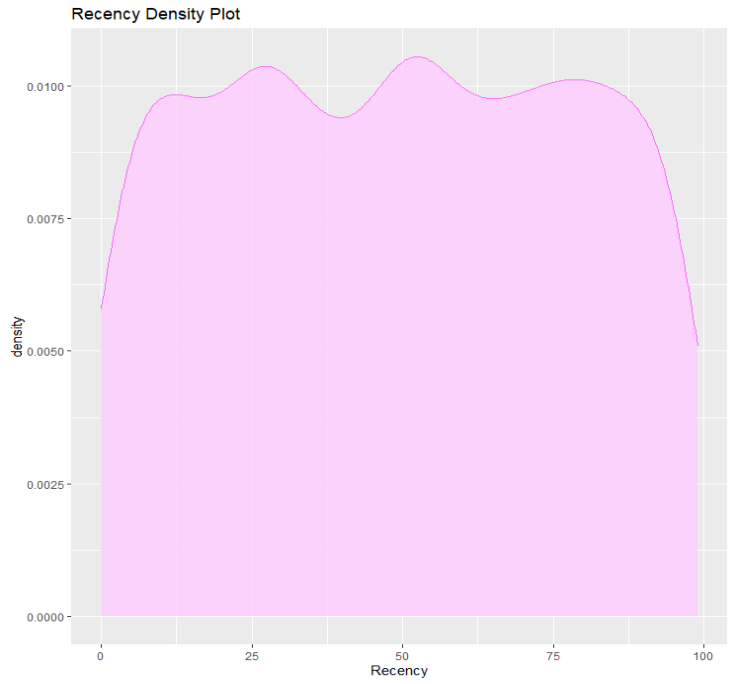
[그림 6. Age Density Plot] Age 변수의 분포를 나타낸 density plot이다. 고객의 대부분이 40-50대에 분포하고 있음을 확인할 수 있다.



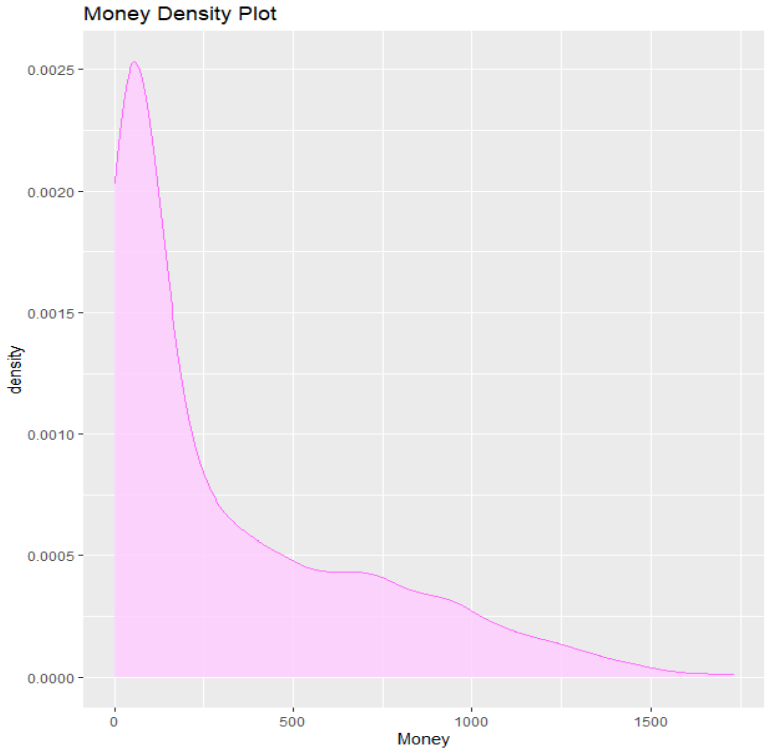
[그림 7. Number of Kids Histogram] 가정 내 아이들의 수를 나타내는 히스토그램이다. 대부분의 가정 내 아이가 존재하지 않으며, 2명인 가정도 1명인 가정에 비해 매우 적은 것을 확인할 수 있다. 또한 3명 이상의 아이가 존재하는 가구는 없는 것을 알 수 있다.



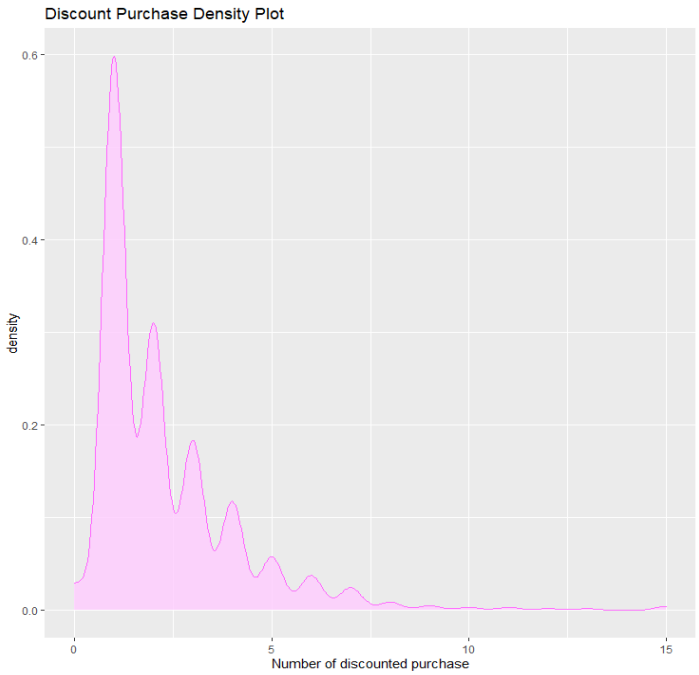
[그림 8. Number of Teens Histogram] 가정 내 청소년의 수를 나타내는 히스토그램이다. 대부분의 가정 내 청소년이 존재하지 않으며, 2명인 가정도 1명인 가정에 비해 매우 적은 것을 확인할 수 있다. 또한 3명 이상의 청소년이 존재하는 가구는 없음을 알 수있다.



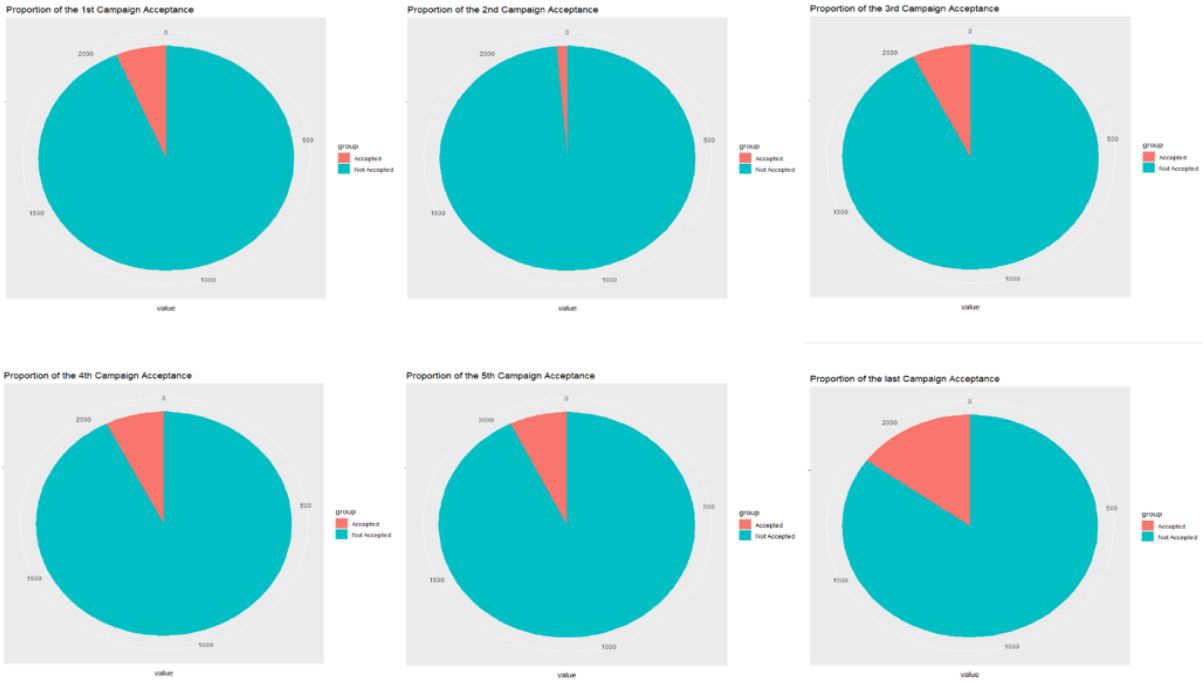
[그림 9. Recency Density Plot] 최근 구매로부터 지난 날짜를 나타내는 Recency 변수의 분포를 나타낸다. Recency의 경우 굉장히 골고루 분포하고 있는 것을 확인할 수 있다.



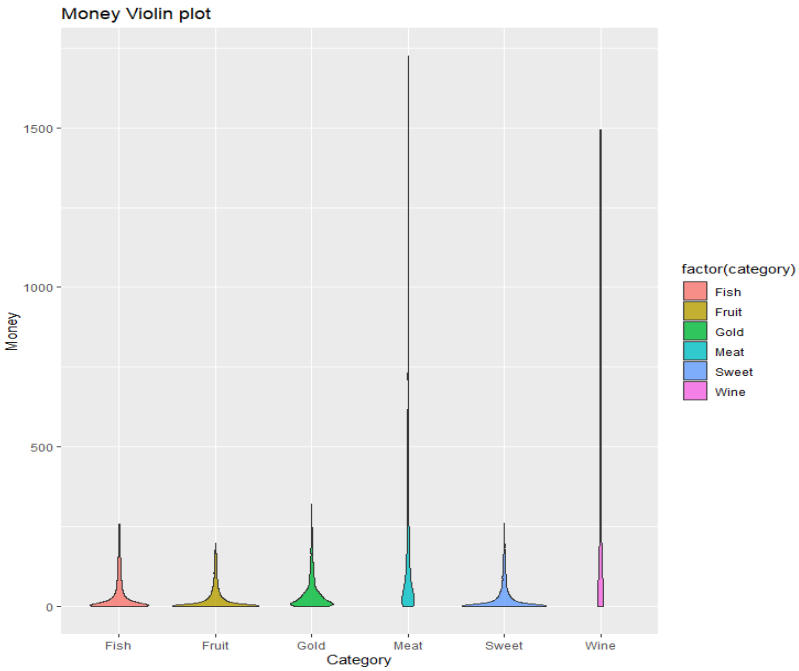
[그림 10. Money Density Plot] 각 가정에서 지출한 비용의 분포를 나타낸다. 오른쪽으로 꼬리가 긴 분포를 띄는 것을 확인할 수 있으며, 전체적으로 소비가 적은 가구가 많으나, 일부 가구의 소비 금액이 매우 큰 것을 확인할 수 있다.



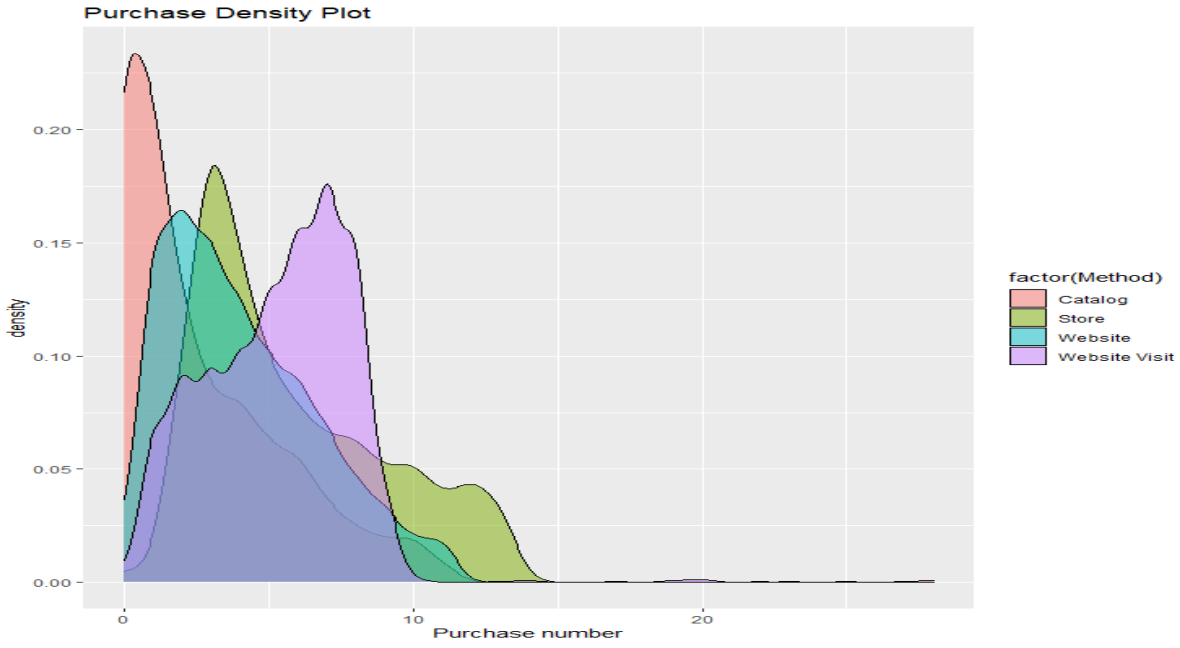
[그림 11. Discount Purchase Density Plot] 할인된 가격으로 제품을 구매한 횟수를 나타내는 Discount Purchase 변수의 분포를 나타낸다. 오른쪽으로 꼬리가 긴 형태를 나타내는 것을 확인할 수 있고, 대부분의 고객은 1-2번의 할인을 받은 경험이 있다는 것을 알 수 있다.



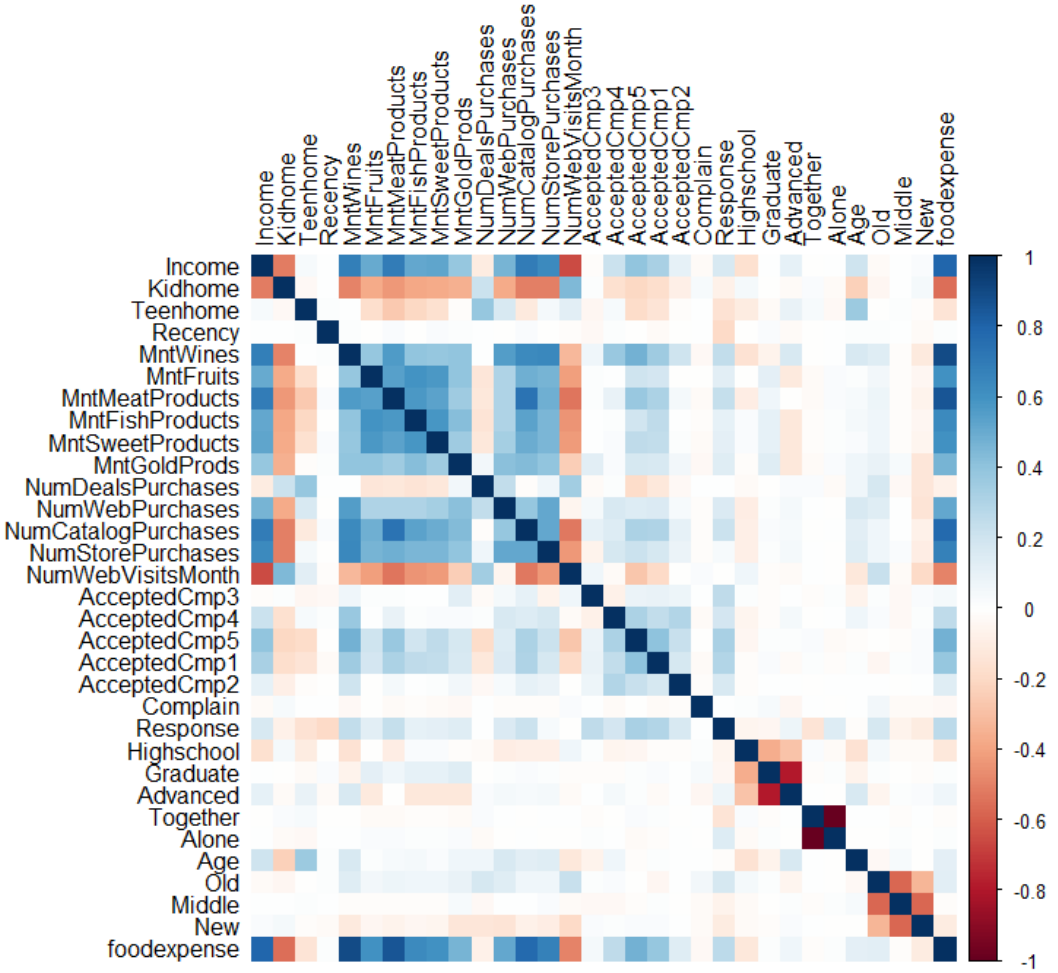
[그림 12. Marketing Campaign Acceptance Pieplot] 첫 번째부터 여섯 번째 마케팅에 참여한 고객의 비율을 나타내는 pieplot이다. 전체적으로 5%의 고객만이 캠페인에 참여하는 것을 확인할 수 있다.



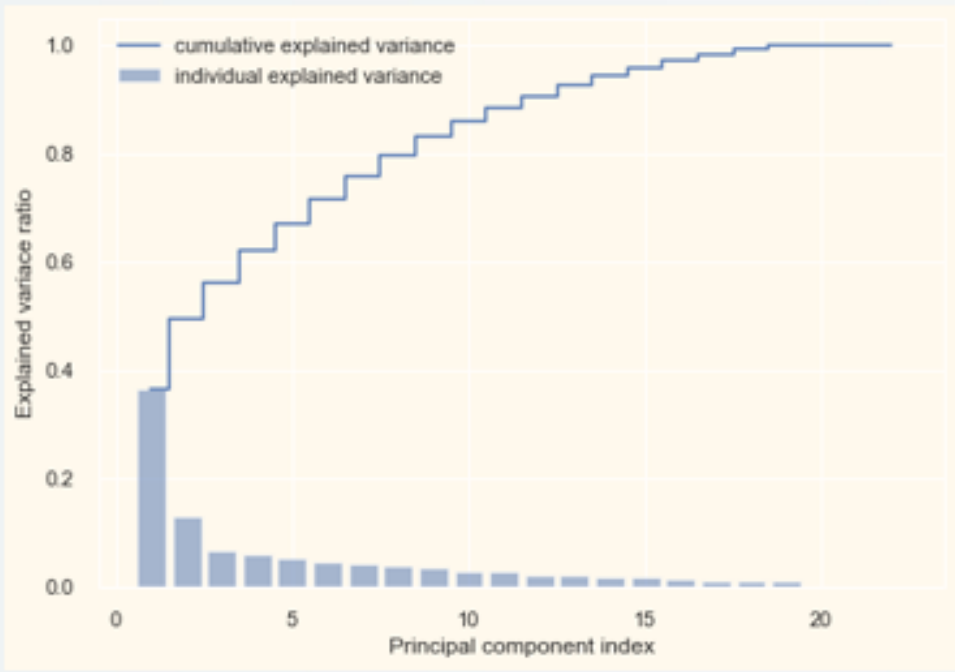
[그림 13. Money Violin Plot] 각 카테고리 별 지출한 금액을 violin plot으로 나타낸 것이다. Fish, Fruit, Gold, Sweet의 경우 대부분의 고객의 소비금액이 매우 적고 좁게 분포하고 있으나, Meat와 Wine의 경우 꼬리가 긴 형태로 고객 별로 소비 금액의 편차가 매우 큰 것을 확인할 수 있다.



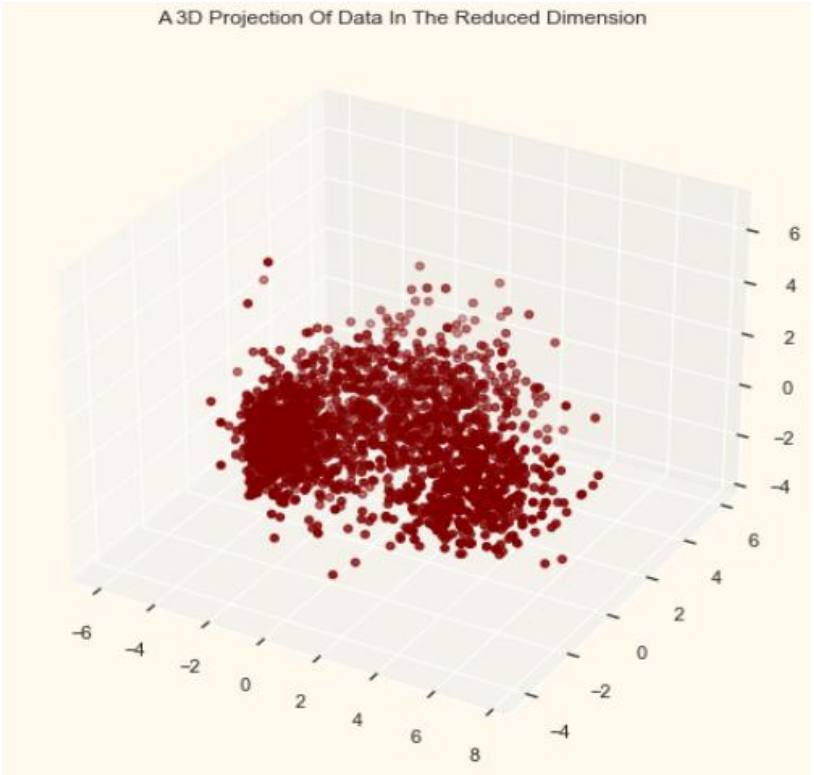
[그림 14. Purchase Density Plot] 카탈로그, 오프라인, 온라인 별 구매 횟수의 분포와 한 달간 웹사이트의 방문 횟수의 분포를 나타낸다. 이를 통해 카탈로그를 이용하여 구매한 고객은 상대적으로 적은 편이며, 오프라인을 통해 구매한 고객이 상대적으로 많은 편인 것을 확인할 수 있다.



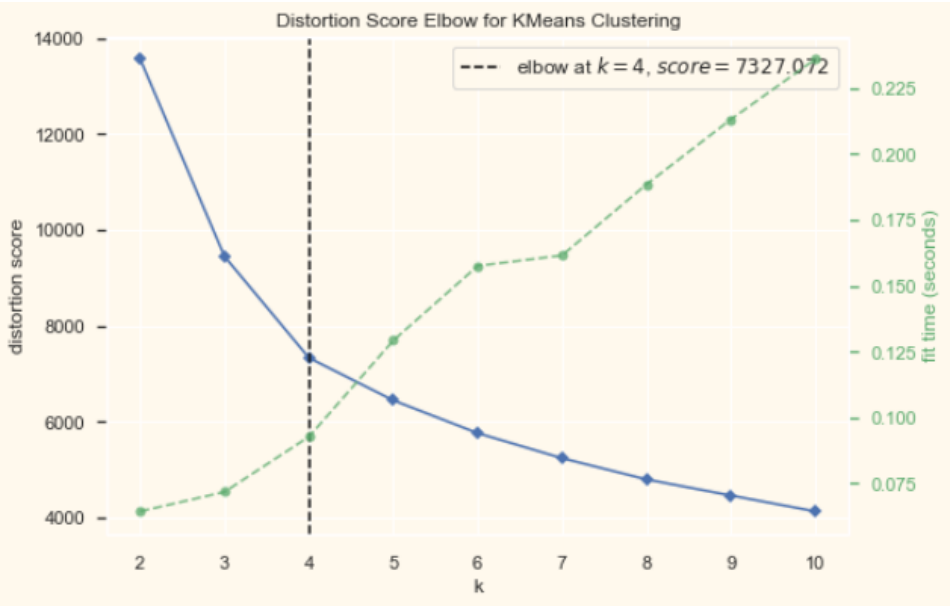
[그림 15. Correlation Plot] 변수들 간의 상관관계를 나타내는 상관관계 플롯이다. 더미 변수를 사용하여 나타낸 변수들을 제외하고는 변수들 간 큰 상관관계가 존재하지 않음을 확인할 수 있다.



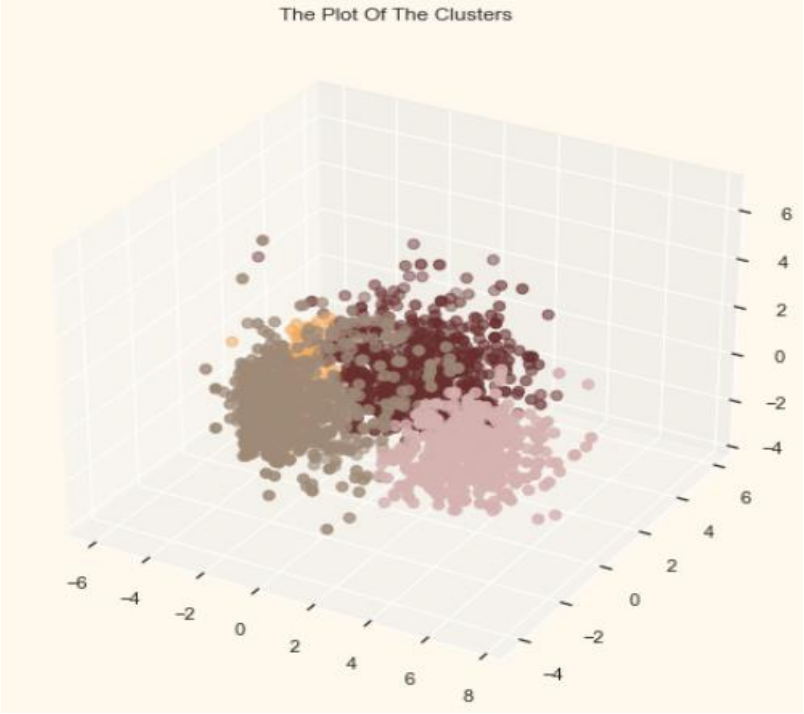
[그림 16. PCA COMPONENT Explained variace ratio] PCA component의 분산 설명량이 4부터 0.1 아래로 내려가는 것을 확인할 수 있다.



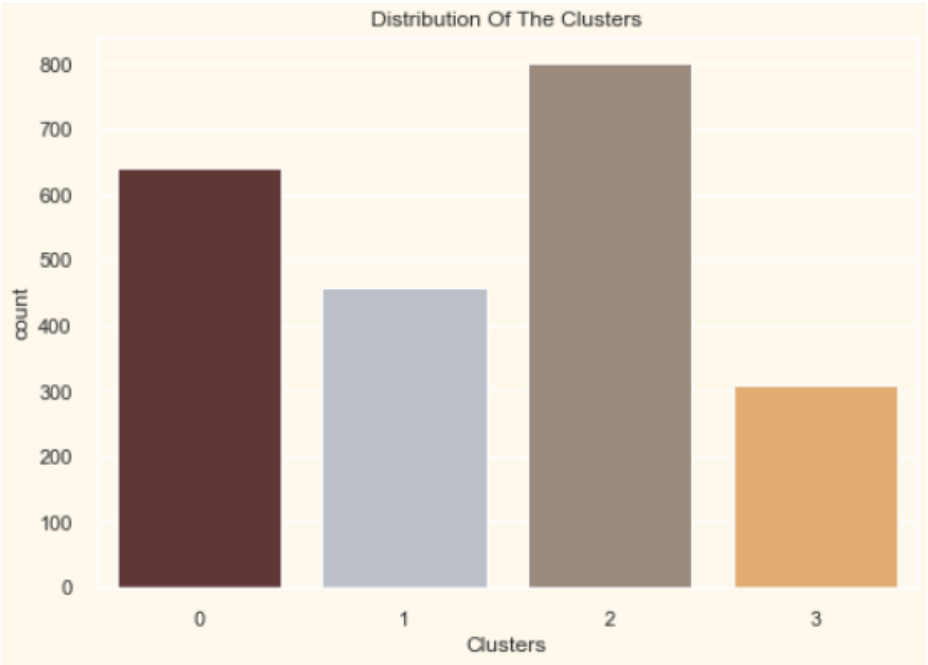
[그림 17. PCA 3D PLOT] PCA 축 상에서 자료가 잘 퍼져 있는 것을 확인할 수 있다.



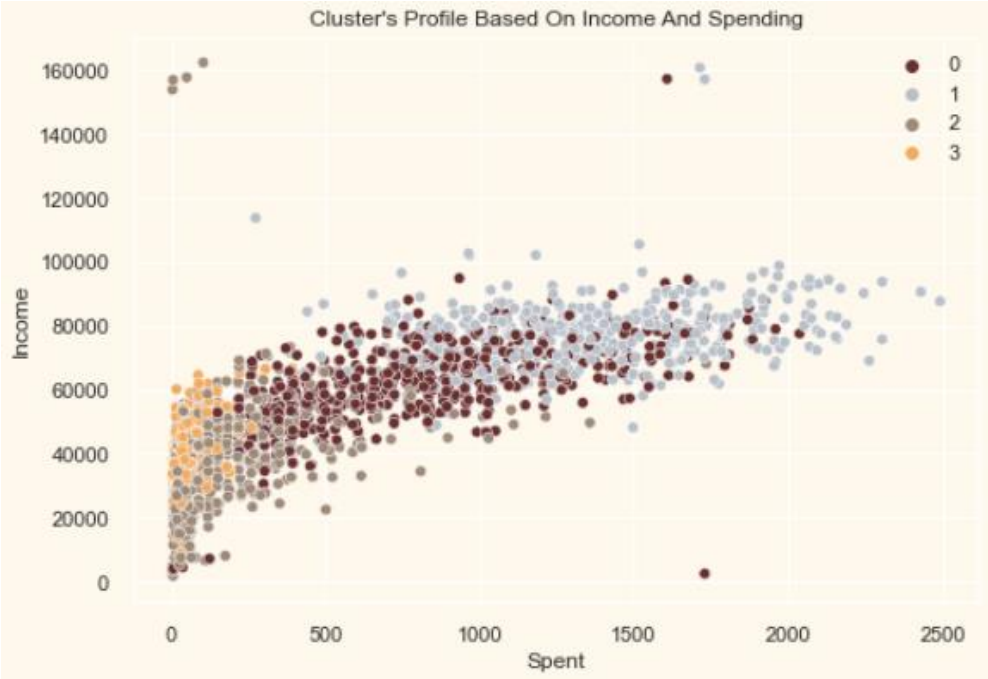
[그림 18. Distortion Score Graph of K-means Clustering] Distortion score를 기준으로 k-means clustering을 한 결과, elbow point가 4로 나타나는 것을 확인할 수 있다.



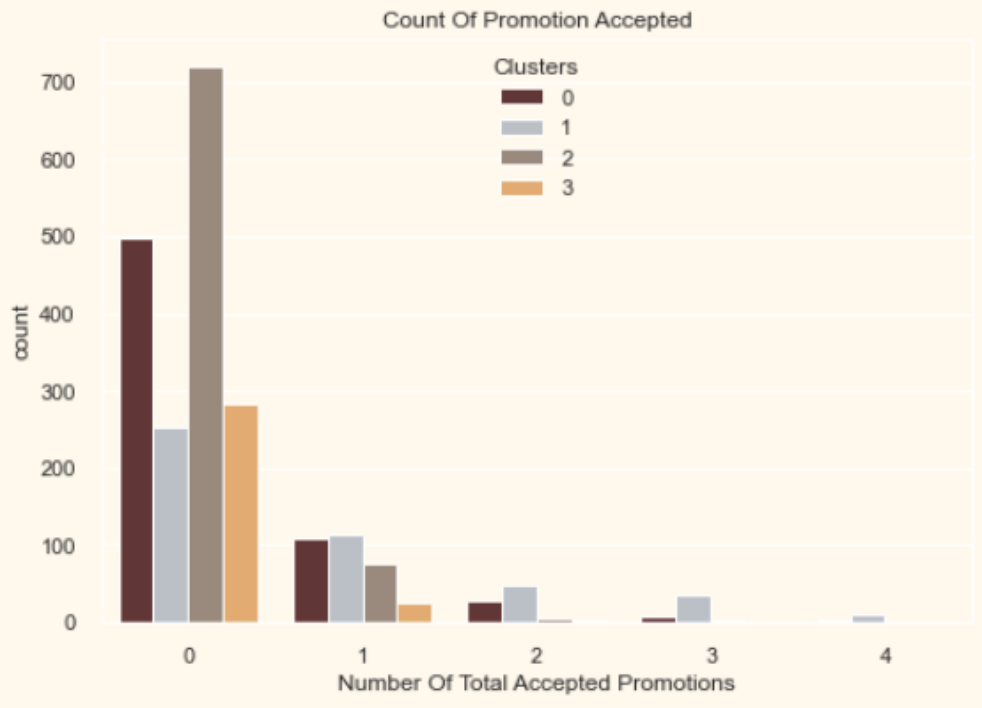
[그림 19. Agglomerative Clustering's cluster on PCA 3D PLOT] PCA 축 상에서 cluster가 잘 분류되고 있음을 확인할 수 있다.



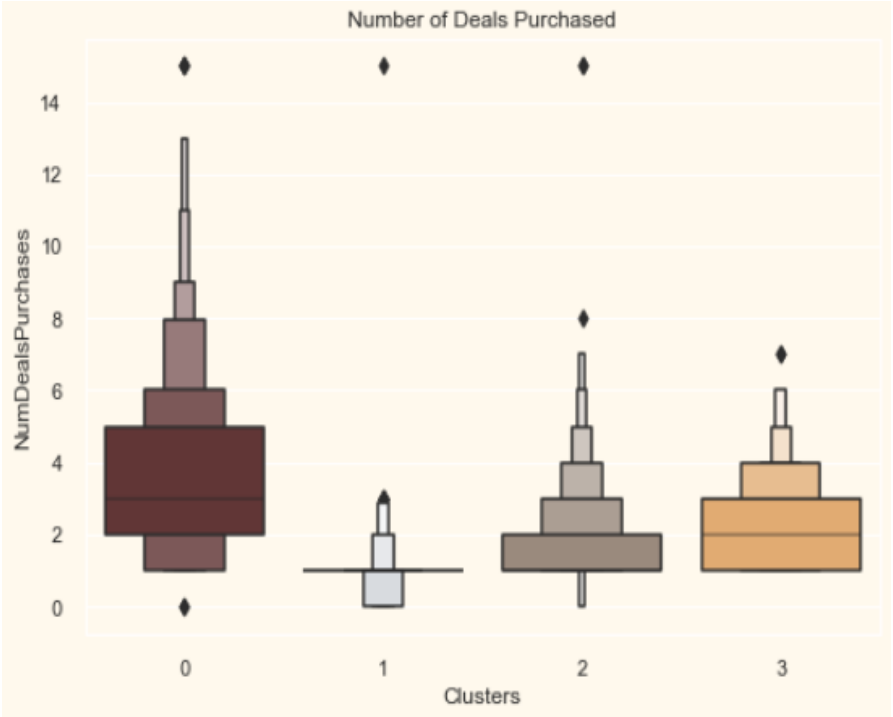
[그림 20. Distribution of The Clusters] 데이터가 cluster별로 골고루 분포되고 있는 것을 확인할 수 있다.



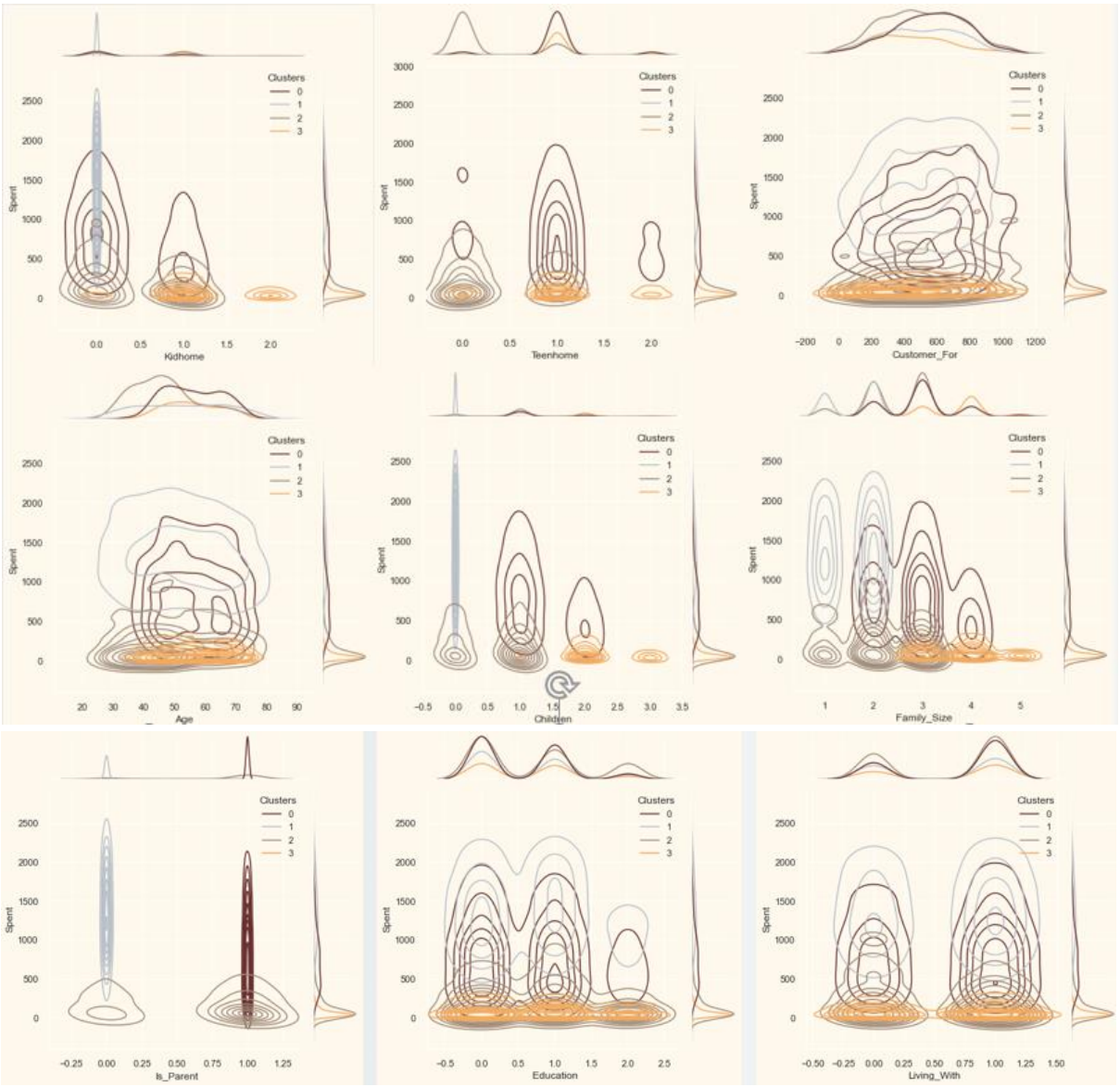
[그림 21. Cluster's Profile Based On Income And Spending] Income과 spending을 축으로 클러스터가 골고루 퍼져있는 것을 확인할 수 있다.



[그림 22. Number Of Total Accepted Promotions] Promotion에 2번 이상 응한 사람은 모든 cluster 내에서 거의 없는 것을 알 수 있다.



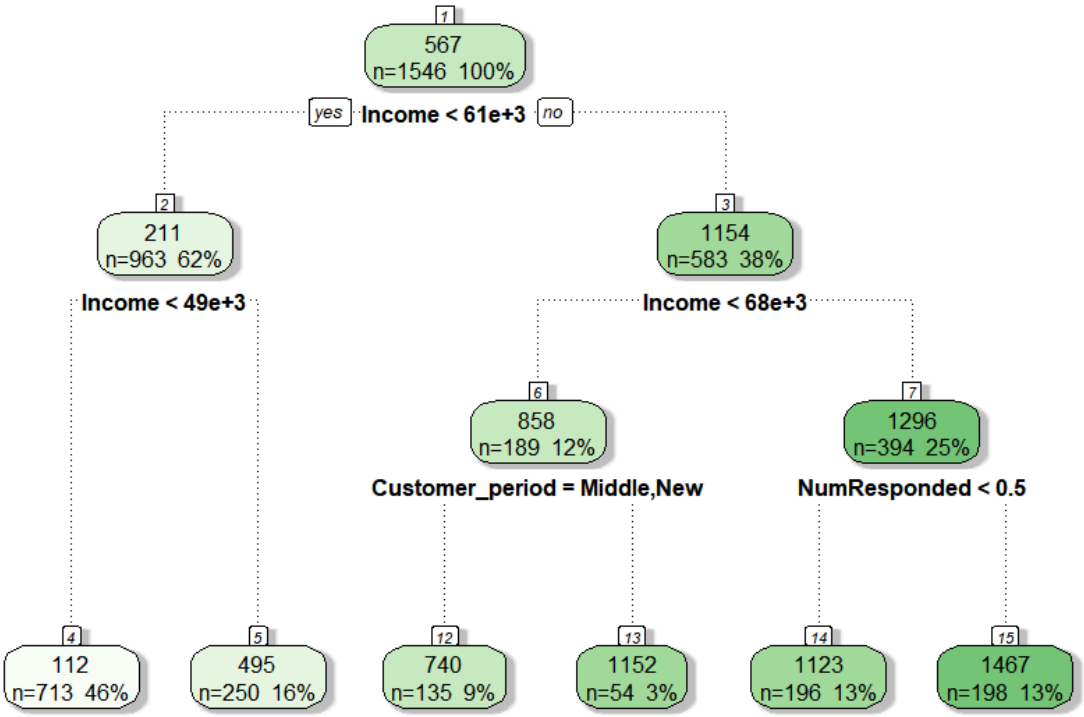
[그림 23. Number of Deals Purchased] Cluster 1에서는 할인을 거의 받지 않았고, cluster 0에서 할인을 제일 많이 받았음을 알 수 있다.



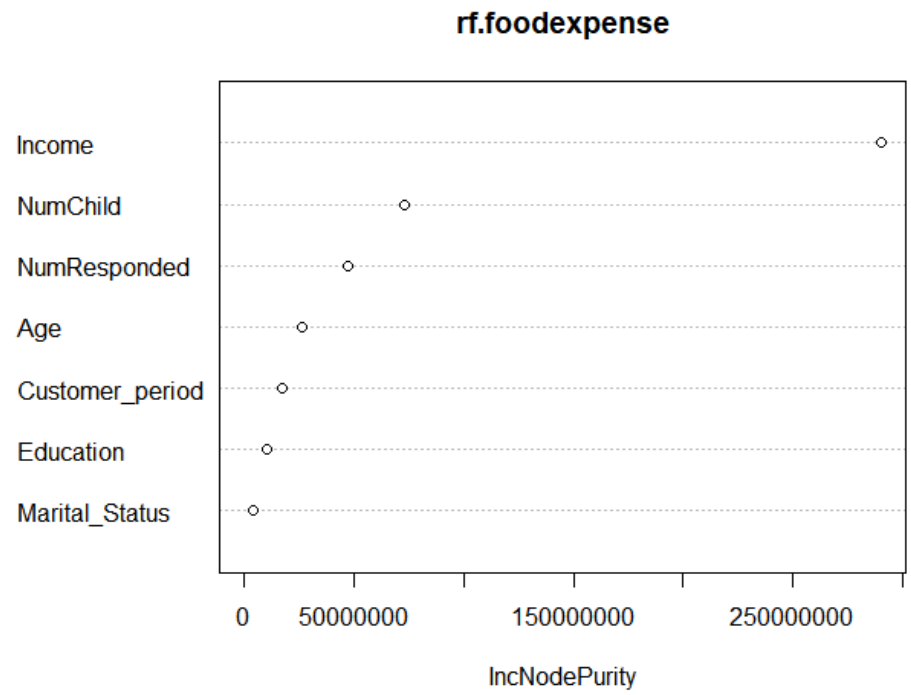
[그림 24. Personal trait-Food Expense Plot] Cluster의 특징을 파악하기 위한 KDE plot이다.

```
Call:
lm(formula = foodexpense ~ NumChild + Customer_period + NumResponded +
    Income, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-2505.89  -171.79   -17.24   160.05  1962.09

Coefficients:
                   Estimate Std. Error t value             Pr(>|t|)
(Intercept)         557.210     11.039  50.478 < 0.0000000000000002 ***
NumChild           -129.002      8.331 -15.484 < 0.0000000000000002 ***
Customer_periodNew -100.765     18.905  -5.330    0.000000112775965 ***
Customer_periodOld  137.736     18.936   7.274    0.000000000000555 ***
NumResponded         93.363      8.392  11.126 < 0.0000000000000002 ***
Income              380.118      8.653  43.930 < 0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 304 on 1540 degrees of freedom
Multiple R-squared:  0.726,     Adjusted R-squared:  0.7251
F-statistic: 816.1 on 5 and 1540 DF,  p-value: < 0.00000000000000022
```

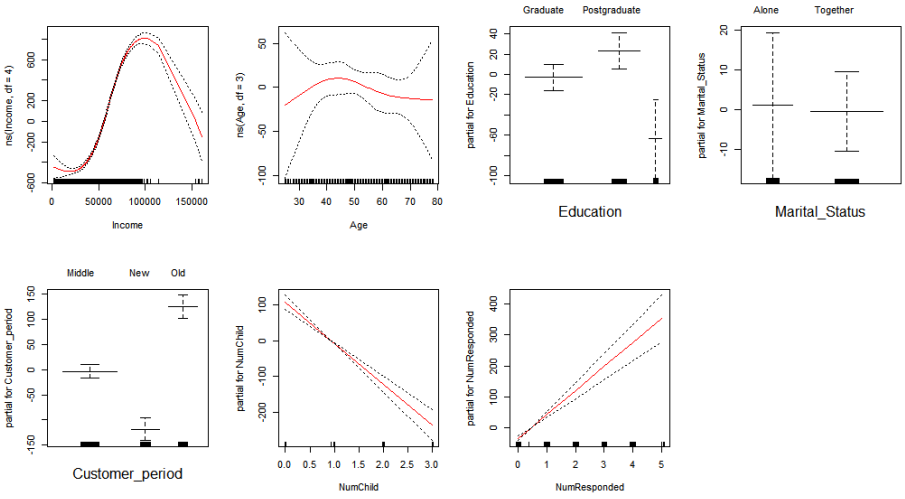
[그림 25. Linear Regression output] Stepwise selection 방식으로 유의하지 않은 변수를 제거한 후 선형 회귀 분석을 진행한 결과이다.



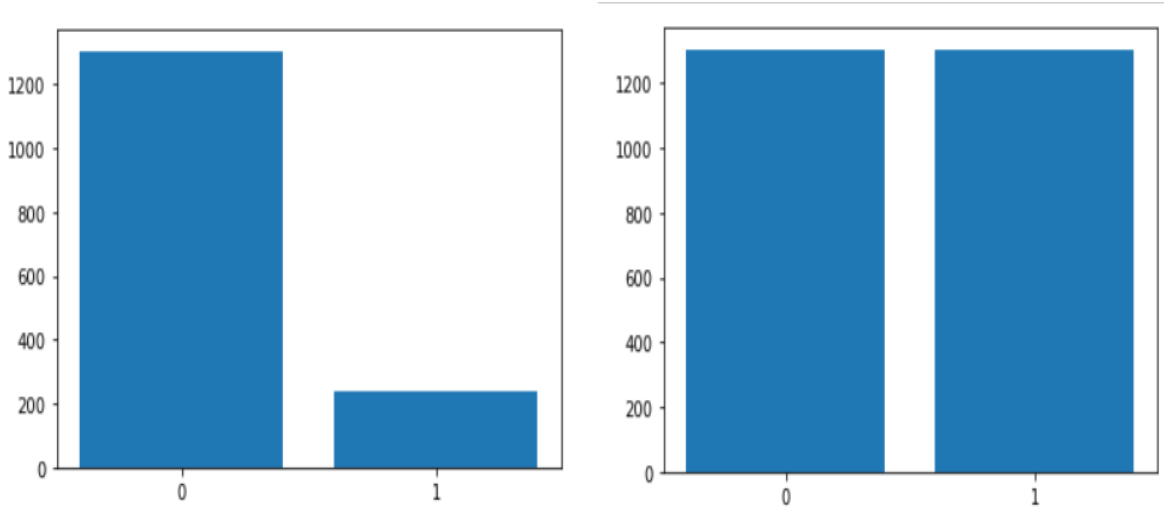
[그림 26. Decision Tree plot] Rpart 패키지의 default cp(=0.01) 값을 활용해 생성한 트리를 시각화한 결과이다. 소득이 가장 영향력 있는 변수로 나타나 첫번째 분리 규칙으로 사용되었다.



[그림 27. Random Forest Variable Importance plot] Random Forest 모델을 적합시킨 후 변수 중요도를 시각화한 결과이다. 소득, 자식 수, 캠페인에 반응한 횟수 등의 순으로 중요한 것으로 나타났다.



[그림 28. Generalized Additive Model 적합 결과] 소득과 나이 변수에 natural spline을 적용한 GAM 모델 적합 결과이다.



[그림 29. SMOTE 사용 결과] 기존의 Train 자료와 SMOTE 적용 이후 반응변수의 분포를 시각화하였다. 반응 변수인 Response 변수의 불균형이 해소된 것을 확인할 수 있다.

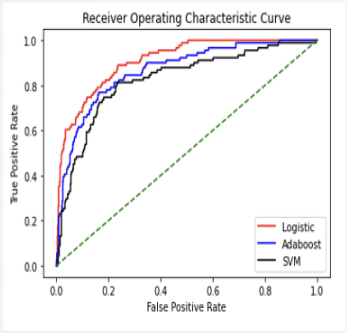
Results: Logit

Model:	Logit	Pseudo R-squared:	0.415
Dependent Variable:	y	AIC:	2159.4432
Date:	2021-12-01 12:26	BIC:	2288.5196
No. Observations:	2610	Log-Likelihood:	-1057.7
Df Model:	21	LL-Null:	-1809.1
Df Residuals:	2588	LLR p-value:	8.8367e-306
Converged:	1.0000	Scale:	1.0000
No. Iterations:	7.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
Recency	-3.5303	0.2276	-15.5117	0.0000	-3.9764	-3.0842
MntWines	-1.9970	0.4717	-4.2332	0.0000	-2.9216	-1.0724
MntMeatProducts	5.5169	0.7431	7.4240	0.0000	4.0604	6.9734
MntFishProducts	-0.8740	0.3789	-2.3063	0.0211	-1.6167	-0.1312
NumDealsPurchases	3.1163	0.5546	5.6194	0.0000	2.0294	4.2033
NumWebPurchases	4.2561	0.8535	4.9865	0.0000	2.5832	5.9290
NumCatalogPurchases	5.3379	0.9610	5.5544	0.0000	3.4543	7.2215
NumStorePurchases	-3.1182	0.3686	-8.4587	0.0000	-3.8407	-2.3957
NumWebVisitsMonth	3.7338	0.8296	4.5008	0.0000	2.1078	5.3597
AcceptedCmp3	2.0839	0.2051	10.1616	0.0000	1.6820	2.4859
AcceptedCmp4	1.3319	0.2333	5.7091	0.0000	0.8746	1.7891
AcceptedCmp5	2.2220	0.2712	8.1929	0.0000	1.6905	2.7536
AcceptedCmp1	0.9045	0.2670	3.3871	0.0007	0.3811	1.4279
AcceptedCmp2	1.3253	0.5653	2.3442	0.0191	0.2173	2.4334
Age	-0.8262	0.2900	-2.8489	0.0044	-1.3946	-0.2578
Living_With	-1.1804	0.1194	-9.8849	0.0000	-1.4145	-0.9464
Is_Parent	-1.1014	0.1926	-5.7190	0.0000	-1.4789	-0.7240
Education_Graduate	1.8253	0.4188	4.3584	0.0000	1.0045	2.6462
Education_Postgraduate	2.5149	0.4205	5.9808	0.0000	1.6907	3.3390
Education_Undergraduate	1.0811	0.4330	2.4967	0.0125	0.2324	1.9298
Dt_Customer_normal	-1.2344	0.1333	-9.2625	0.0000	-1.4956	-0.9732
Dt_Customer_recent	-2.2405	0.2034	-11.0144	0.0000	-2.6392	-1.8418

[그림 30. Logistic Regression의 회귀계수 결과] Backward selection을 사용한 변수 선택 이후, Logistic Regression을 적합한 결과이다.

	Logistic Regression	AdaBoost	SVM	
			all	selection
CV AUC	0.8924	0.9826	**0.9867**	0.9818
Test AUC	**0.9099**	0.7008	0.6542	0.6450



[그림 31. Cross validation mean AUC와 Test AUC 결과] 분류 모형 별 ROC Curve Plot과 AUC값을 정리한 표이다. CV AUC를 기준으로 하였을 때는 SVM이, test AUC를 기준으로 하였을 때는 Logistic Regression의 성능이 가장 좋은 것을 확인할 수 있다.

cut-off value: 0.5

	precision	recall	f1-score
0	0.96	0.83	0.89
1	0.43	**0.79**	0.56

Test Accuracy : 약 0.8281

cut-off value: 0.7

	precision	recall	f1-score
0	0.94	0.92	0.93
1	0.58	0.66	0.62

Test Accuracy : 약 0.8884

cut-off value: 0.9

	precision	recall	f1-score
0	0.91	0.99	0.95
1	**0.83**	0.42	0.55

Test Accuracy : 약 0.9080

[그림 32. Logistic Regression의 cut-off value 변경에 따른 confusion matrix 결과] Test AUC score가 가장 높았던 Logistic Regression에 대해 얻은 결과로, Response 변수가 1일 때, 즉 최근 캠페인에 응답한 것에 대한 recall 값은 cut-off value가 0.5일 때 가장 높은 것을 확인할 수 있고, precision 값은 cut-off value가 0.9일 때 가장 높은 것을 확인할 수 있다.

- **Codes for Data Preprocessing (by R)**

```
mkt=read.csv("marketing_campaign.csv",header=TRUE,sep="₩t")

head(mkt)

str(mkt)

library(tidyverse)

library(ggplot2)

library(dplyr)

mkt%>%gather(key="key",value="val")%>%mutate(isna=is.na(val))%>%group_by(key)%>%

  summarise(na=sum(isna))

mkt=mkt%>%filter(Income!=is.na(Income)

table(mkt$Marital_Status)

table(mkt$Education)

summary(mkt$Income) #Min 1730, Q1 35303 Median 51382 Mean 52247 Q3 68522 Max
666666

sort(mkt$Income,decreasing = TRUE)

mkt=mkt%>%mutate(Dt_Customer=as.Date(Dt_Customer,"%d-%m-%Y"))

mkt=mkt[,-c(27,28)]

which.min(mkt$Income)

mkt=mkt[-2210,]

mkt%>%group_by(Education)%>%summarise(mean(Income))

mkt=mkt%>%mutate(Highschool=ifelse(Education=="Basic" | Education=="2n Cycle",1,0))

mkt=mkt%>%mutate(Graduate=ifelse(Education=="Graduation",1,0))

mkt=mkt%>%mutate(Advanced=ifelse(Education=="Master" | Education =="PhD",1,0))

hist(mkt$Year_Birth)
```

```
sort(mkt$Year_Birth,decreasing=FALSE)

mkt=mkt%>%filter(!Year_Birth %in% c(1893,1899,1900))

mkt=mkt%>%filter(!Marital_Status %in% c("YOLO","Absurd"))

mkt=mkt%>%mutate(Together=ifelse(Marital_Status=="Together"|
Marital_Status=="Married",1,0))

mkt=mkt%>%mutate(Alone=ifelse(Together==0,1,0))

mkt=mkt%>%mutate(Age=2021-Year_Birth)

summary(mkt$Dt_Customer)

q1=as.Date("2013-01-16")

q3=as.Date("2013-12-31")

mkt=mkt%>%mutate(Old=ifelse(Dt_Customer <= q1,1,0))

mkt=mkt%>%mutate(Middle=ifelse(Dt_Customer <= q3 & Dt_Customer> q1,1,0))

mkt=mkt%>%mutate(New=ifelse(Dt_Customer>q3,1,0))

mkt=mkt%>%select(-c(Marital_Status,Education,Dt_Customer,ID,Year_Birth))

mkt=mkt%>%mutate(foodexpense=MntWines+MntFruits+MntMeatProducts+MntFishProdu
cts+MntSweetProducts)

write.csv(mkt,"mkt.csv")
```

- **Codes for EDA (by R)**

```
mkt=read.csv("mkt.csv",header=TRUE)

  library(dplyr)

  library(tidyverse)

  library(ggplot2)

  mkt=mkt[,-1]

  str(mkt)

  glimpse(mkt)
```

```
summary(mkt)

ggplot(mkt,aes(y=Income))+geom_boxplot(fill='#CCCCFF',col='#6666FF')+
labs(title="Income Boxplot",y="Income")

ggplot(mkt,aes(Age))+geom_density(alpha=0.8,fill='#FFCCFF')+labs(title="Age Density
Plot",x="Age")


ggplot(mkt,aes(Kidhome))+geom_histogram(alpha=0.8,fill='#FFCCFF',bins=3,col='#FF66FF')+l
abs(title="Number of Kids",x="Number of Kids at home")


ggplot(mkt,aes(Teenhome))+geom_histogram(alpha=0.8,fill='#CC99FF',bins=3,col='#9966CC'
)+labs(title="Number of Teens",x="Number of Teens at home")


ggplot(mkt,aes(Recency))+geom_density(alpha=0.8,fill='#FFCCFF',col='#FF66FF')+labs(title="
Recency Density Plot",x="Recency")

money=as.matrix(c(mkt$MntWines,mkt$MntFruits,mkt$MntMeatProducts,mkt$MntFishProdu
cts,mkt$MntSweetProducts,mkt$MntGoldProds),ncol=1)

category=rep(c("Wine","Fruit","Meat","Fish","Sweet","Gold"),each=2208)

moneyplot=cbind(money,category)

moneyplot=as.data.frame(moneyplot)

ggplot(moneyplot,aes(category,money))+geom_violin(

  aes(fill=factor(category)),alpha=0.8)+

 labs(title="Money Violin plot",x="Category",y="Money")

mktori=read.csv("marketing_campaign.csv",header=TRUE,sep="₩t")

ggplot(mktori)+geom_bar(aes(x=Marital_Status),fill='#6666FF')+labs(title="Marital Status
Histogram",x="Marital Status",y="Count")

purchase=as.matrix(c(mkt$NumWebPurchases,mkt$NumCatalogPurchases,mkt$NumStorePur
chases,mkt$NumWebVisitsMonth),ncol=1)

Method=rep(c("Website","Catalog","Store","Website Visit"),each=2208)
```

```
purchaseplot=cbind(purchase,Method)

purchaseplot=as.data.frame(purchaseplot)


ggplot(purchaseplot,aes(purchase))+geom_density(aes(fill=factor(Method)),alpha=0.5)+labs(t
itle="Purchase Density Plot",x="Purchase number")

#Purchase density plot

mkt1=data.frame(group=c("Accepted","Not Accepted"),value=c(141,2067))

mkt2=data.frame(group=c("Accepted","Not Accepted"),value=c(30,2178))

mkt3=data.frame(group=c("Accepted","Not Accepted"),value=c(163,2045))

mkt4=data.frame(group=c("Accepted","Not Accepted"),value=c(164,2044))

mkt5=data.frame(group=c("Accepted","Not Accepted"),value=c(160,2048))

res=data.frame(group=c("Accepted","Not Accepted"),value=c(331,1877))

comp=data.frame(group=c("Yes","No"),value=c(20,2188))


ggplot(mkt1,aes(x="",y=value,fill=group))+geom_bar(width=1,stat="identity")+coord_polar("
y")+labs(title="Proportion of the 1st Campaign Acceptance",x=NULL)


ggplot(mkt2,aes(x="",y=value,fill=group))+geom_bar(width=1,stat="identity")+coord_polar("
y")+labs(title="Proportion of the 2nd Campaign Acceptance",x=NULL)


ggplot(mkt3,aes(x="",y=value,fill=group))+geom_bar(width=1,stat="identity")+coord_polar("
y")+labs(title="Proportion of the 3rd Campaign Acceptance",x=NULL)


ggplot(mkt4,aes(x="",y=value,fill=group))+geom_bar(width=1,stat="identity")+coord_polar("
y")+labs(title="Proportion of the 4th Campaign Acceptance",x=NULL)


ggplot(mkt5,aes(x="",y=value,fill=group))+geom_bar(width=1,stat="identity")+coord_polar("
y")+labs(title="Proportion of the 5th Campaign Acceptance",x=NULL)
```

```
ggplot(res,aes(x="",y=value,fill=group))+geom_bar(width=1,stat="identity")+coord_polar("y")
+labs(title="Proportion of the last Campaign Acceptance",x=NULL)


ggplot(comp,aes(x="",y=value,fill=group))+geom_bar(width=1,stat="identity")+coord_polar("
y")+labs(title="Proportion of Complain Experienced",x=NULL)

ggplot(mkt,aes(NumDealsPurchases))+geom_density(alpha=0.8,fill='#FFCCFF',col='#FF66FF')
+labs(title="Discount Purchase Density Plot",x="Number of discounted purchase")

ggplot(mktori)+geom_bar(aes(x=Education),fill='#6666FF')+labs(title="Education
Histogram",x="Education",y="Count")

library(corrplot)

  correlation=cor(mkt)

  corrplot(correlation,method ="color",tl.col="black")


mkt=mkt%>%mutate(totalmoney=MntFishProducts+MntFruits+MntMeatProducts+MntGold
Prods+MntSweetProducts+MntSweetProducts)


ggplot(mkt,aes(x=totalmoney))+geom_density(alpha=0.8,fill='#FFCCFF',col='#FF66FF')+labs(ti
tle="Money Density Plot",x="Money")
```

- **Codes for Clustering (전처리 이후의 Python Code를 작성)**

```
## preprocessing the data to perform clustering operations.

### The following steps are applied to preprocess the data:

# - Label encoding the categorical features ( clustering 적용 위해 )

# - Scaling the features using the standard scaler ( PCA는 scale-variant! 그래서 표준화 )

# - Creating a subset dataframe for dimensionality reduction ( multicollinearity Issue 발생
막기 위해 )
```

```
#Get list of categorical variables

s = (data.dtypes == 'object')

object_cols = list(s[s].index)

print("Categorical variables in the dataset:", object_cols)

#Label Encoding the object dtypes.

LE=LabelEncoder()

for i in object_cols:

    data[i]=data[[i]].apply(LE.fit_transform)

print("All features are now numerical")

# creating a copy of data

ds = data.copy()


# creating a subset of dataframe by dropping the features on deals accepted and promotions

cols_del=['AcceptedCmp3','AcceptedCmp4','AcceptedCmp5',
'AcceptedCmp1','AcceptedCmp2', 'Complain', 'Response']

ds = ds.drop(cols_del, axis=1)

# Scaling

scaler = StandardScaler()

scaled_ds = pd.DataFrame(scaler.fit_transform(ds), columns=ds.columns)

print("All features are now scaled")

#Scaled data to be used for reducing the dimensionality

print("Dataframe to be used for further modelling:")

scaled_ds.head()
```

```
## DIMENSIONALITY REDUCTION ( The higher the number of features, the harder it is to work with it. Many of these features are correlated, and hence redundant.)

## - Dimensionality reduction with PCA

## - Plotting the reduced dataframe

## - Dimensionality reduction with PCA

cov_mat=np.cov(scaled_ds.T)

eigen_vals,eigen_vecs=np.linalg.eig(cov_mat)

print('Eigenvalue of X variables ! ₩n %s'% eigen_vals)

## PCA로 설명되는 분산을 계산하고 plotting!!

tot=sum(eigen_vals)

var_exp=[(i/tot) for i in sorted(eigen_vals,reverse=True)]

cum_var_exp=np.cumsum(var_exp)

plt.bar(range(1,23), var_exp, alpha=0.5, label='individual explained variance')

plt.step(range(1,23),cum_var_exp, where='mid', label='cumulative explained variance')

plt.ylabel('Explained variace ratio')

plt.xlabel('Principal component index')

plt.legend(loc='best')

# 분산 설명량이 0.1 이하로 감소하는 3차원을 기준으로 축소

pca = PCA(n_components=3)

pca.fit(scaled_ds)

PCA_ds = pd.DataFrame(pca.transform(scaled_ds), columns=(["col1","col2","col3"]))

PCA_ds.describe().T

#A 3D Projection Of Data In The Reduced Dimension

x =PCA_ds["col1"]

y =PCA_ds["col2"]
```

```
z =PCA_ds["col3"]

#To plot

fig = plt.figure(figsize=(10,8))

ax = fig.add_subplot(111, projection="3d")

ax.scatter(x,y,z, c="maroon", marker="o" )

ax.set_title("A 3D Projection Of Data In The Reduced Dimension")

plt.show()

## CLUSTERING ( Distortion score 기준으로  clustering의 차원 결정! )

## - Elbow Method to determine the number of clusters to be formed

##- Clustering via Agglomerative Clustering

## - Examining the clusters formed via scatter plot

print('Elbow Method to determine the number of clusters to be formed:')

Elbow_M = KElbowVisualizer(KMeans(), k=10, metric = 'distortion')

Elbow_M.fit(PCA_ds)

Elbow_M.show()


# Initiating the Agglomerative Clustering model -> 계층적 클러스터링

# (Recursively merges the pair of clusters that minimally increases a given linkage distance.)

# 계층적 클러스터링 사용 이유? cluster의 거리 측정 기준인 linkage 선택 가능! 우리는 어느 정도의 크기가 있는 cluster 집단을 만들고 싶었다.

# 그래서 ward linkage 사용:

# Ward linkage?

# 모든 클러스터 내의 분산을 가장 작게 증가시키는 두 클러스터를 합침, 크기가 비교적 비슷한 클러스터가 만들어짐

# 선택 이유 : 일반적으로 가장 많이 사용되는 linkage이다. / 어느 정도의 크기가 있는
```

cluster 집단을 만들어야 의미가 있다고 판단.

```
AC = AgglomerativeClustering(n_clusters=4, linkage='ward')
# fit model and predict clusters
yhat_AC = AC.fit_predict(PCA_ds)
PCA_ds["Clusters"] = yhat_AC
#Adding the Clusters feature to the orignal dataframe.
data["Clusters"]= yhat_AC


#Plotting the clusters
fig = plt.figure(figsize=(10,8))
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(x, y, z, s=40, c=PCA_ds["Clusters"], marker='o', cmap = cmap )
ax.set_title("The Plot Of The Clusters")
plt.show()


#Plotting countplot of clusters
pal = ["#682F2F","#B9C0C9", "#9F8A78","#F3AB60"]
pl = sns.countplot(x=data["Clusters"], palette= pal)
pl.set_title("Distribution Of The Clusters")
plt.show()


# Income vs spending plot shows the clusters pattern
# group 0: high spending & average income
```

```
# group 1: high spending & high income
# group 2: low spending & low income
# group 3: high spending & low income


pl = sns.scatterplot(data = data, x=data["Spent"], y=data["Income"],
                     hue=data["Clusters"], palette= pal)
pl.set_title("Cluster's Profile Based On Income And Spending")
plt.legend()
plt.show()



# Creating a feature to get a sum of accepted promotions
data["Total_Promos"] = data["AcceptedCmp1"]+ data["AcceptedCmp2"]+ data["AcceptedCmp3"]+ data["AcceptedCmp4"]+ data["AcceptedCmp5"]


# Plotting count of total campaign accepted.
plt.figure()
pl = sns.countplot(x=data["Total_Promos"],hue=data["Clusters"], palette= pal)
pl.set_title("Count Of Promotion Accepted")
pl.set_xlabel("Number Of Total Accepted Promotions")
plt.show()


# Plotting the number of deals purchased
# Unlike campaigns, the deals offered did well. It has best outcome with cluster 0 and cluster 3.
```

```
# However, our star customers cluster 1 are not much into the deals. Nothing seems to attract cluster 2 overwhelmingly


plt.figure()

pl=sns.boxenplot(y=data["NumDealsPurchases"],x=data["Clusters"], palette= pal)

pl.set_title("Number of Deals Purchased")

plt.show()

## spent와 Personal_list 안에 있는 변수들을 축으로 두고 jointplot -> cluster의 특징 찾아내기!

Personal_list = ["Kidhome","Teenhome","Customer_For",

                 "Age", "Children", "Family_Size",

                 "Is_Parent", "Education","Living_With"]

fig = plt.figure(figsize=(24, 9*3))

for i, col in enumerate(Personal_list):

    plt.figure()

    sns.jointplot(x=data[col], y=data["Spent"], hue = data["Clusters"], kind="kde", palette=pal)

    plt.show()
```

- **Codes for Regression (by R)**

```
mkt=read.csv("marketing_campaign.csv",header=TRUE,sep="₩t")

library(tidyverse)

library(ggplot2)

library(dplyr)

library(caret)
```

```
#income 결측치 제거

mkt%>%gather(key="key",value="val")%>%mutate(isna=is.na(val))%>%group_by(key)%>%summarise(na=sum(isna))%>%print(n=29)

mkt=mkt%>%filter(Income!=is.na(Income))

#Z_CostContact, Z_Revenue 열 삭제

mkt=mkt[,-c(27,28)]

#Income 이상치 제거

summary(mkt$Income)

boxplot(mkt$Income)

mkt=mkt[!(mkt$Income==666666),]

#Education 범주 3개로

mkt$Education = ifelse((mkt$Education=='Basic') | (mkt$Education=='2n Cycle'), 'Undergraduate', ifelse(mkt$Education=='Graduation', 'Graduate', 'Postgraduate'))

table(mkt$Education)

#age 변수

mkt=mkt%>%mutate(Age=2021-Year_Birth)

boxplot(mkt$Age)

mkt=mkt%>%filter(!Year_Birth %in% c(1893,1899,1900))

#marital_state 처리

mkt=mkt%>%filter(!Marital_Status %in% c("YOLO","Absurd"))

mkt$Marital_Status = ifelse(mkt$Marital_Status=="Together" | mkt$Marital_Status=="Married",'Together','Alone')

table(mkt$Marital_Status)

#dt_customer

mkt=mkt%>%mutate(Dt_Customer=as.Date(Dt_Customer,"%d-%m-%Y"))
```

```
summary(mkt$Dt_Customer)

q1=as.Date("2013-01-16")

q3=as.Date("2013-12-31")

mkt$Customer_period = ifelse(mkt$Dt_Customer <= q1,'Old', ifelse(mkt$Dt_Customer <= q3, 'Middle', 'New'))

table(mkt$Customer_period)

#식자재소비

mkt=mkt%>%mutate(foodexpense=MntWines+MntFruits+MntMeatProducts+MntFishProducts+MntSweetProducts)

#필요없는 열 제거

mkt=mkt%>%dplyr::select(-c(Dt_Customer,ID,Year_Birth))

str(mkt)

mkt$Education = as.factor(mkt$Education)

mkt$Marital_Status = as.factor(mkt$Marital_Status)

mkt$Customer_period = as.factor(mkt$Customer_period)


#########################################################################

### NumChild(자식수), NumResponded(마케팅오퍼에 반응한 횟수) 파생변수 생성

mkt = mkt%>%mutate(NumChild=Kidhome+Teenhome)

mkt = mkt%>%mutate(NumResponded=AcceptedCmp1+AcceptedCmp2+AcceptedCmp3+AcceptedCmp4+AcceptedCmp5+Response)

hist(mkt$NumResponded)


#########################################################################

### foodexpense에 나이, 소득, 학력 등과 같은 인적정보가 어떻게 영향을 미치는지
```

```
### 알아보고자함

mydata <- mkt%>%dplyr::select(c(Age, Education, Marital_Status, NumChild, Customer_period, NumResponded, Income, foodexpense))

head(mydata)

str(mydata)


#########################################################################

### Linear_regression

options(scipen = 100)

#train_test_set_split

set.seed(0)

train.index = sample(1:nrow(mydata), round(0.7*nrow(mydata)))

train = mydata[train.index,] #train data

test = mydata[-train.index,] #test data

#scaling

cols = c('Age', 'NumChild', 'NumResponded', 'Income')

pre_proc_val <- preProcess(train[,cols], method = c("center", "scale"))

train[,cols] = predict(pre_proc_val, train[,cols])

test[,cols] = predict(pre_proc_val, test[,cols])

head(train)

full_model <- lm(foodexpense~., data=train)

summary(full_model)

fit.reg = lm(foodexpense ~ ., data = train)

fit.step.reg = step(fit.reg, direction="both", trace=FALSE) #Stepwise variable selection

summary(fit.step.reg)
```

```
#Cross validation
V = 10 #V-fold CV
mse.train = 0; mse.test = 0
set.seed(0)
id = sample(1:V, nrow(train), replace = T)
for(i in 1:V) {
  print(i)
  ## Data partitioning
  val.index = which(id==i)
  mydata.train = train[-val.index,] #train data
  mydata.test   = train[val.index,] #test data
  ## Fitting
  fit.reg = lm(foodexpense ~ ., data = train)
  fit.step.reg = step(fit.reg, direction="both", trace=FALSE) #Stepwise variable selection
  ## Predicting and Evaluating
  yhat.reg = predict(fit.step.reg, newdata=mydata.test, type="response")
  mse.test = mse.test + mean((mydata.test$foodexpense - yhat.reg)^2)   # MSE
}
cv.mse.test_lm   = mse.test/V;   cv.mse.test_lm   # test CV MSE


#test MSE
fit.reg = lm(foodexpense ~ ., data = train)
fit.step.reg = step(fit.reg, direction="both", trace=FALSE) #Stepwise variable selection
yhat_lm = predict(fit.step.reg, newdata=test)
```

```
test_mse_lm = mean((test$foodexpense-yhat_lm)^2)

test_mse_lm



##############################################

###Decision Tree

train = mydata[train.index,] #train data

test = mydata[-train.index,] #test data scale되지 않은 상태로 사용

library(tree)

library(randomForest)

#set.seed(0)

#tree.fit <- tree(foodexpense ~., data=train)

#summary(tree.fit)

#plot(tree.fit)

#text(tree.fit, pretty = 0)

library(rpart)          # Popular decision tree algorithm

install.packages('rattle')

library(rattle)         # Fancy tree plot

library(rpart.plot)             # Enhanced tree plots

library(RColorBrewer)           # Color selection for fancy tree plot

install.packages('party')

library(party)                  # Alternative decision tree algorithm

library(caret)



tree.1 <- rpart(foodexpense~., data=train, control=rpart.control(minsplit=20, cp=0))
#maximum tree
```

```
plot(tree.1)
text(tree.1)


## Pruning the tree and selecting the tree with min error
tmp = printcp(tree.1)
k = which.min(tmp[,"xerror"])
k
cp.tmp = tmp[k,"CP"]
fit.pruned = prune(tree.1, cp=cp.tmp)
plot(fit.pruned, margin = 0.1);text(fit.pruned, use.n=TRUE)
fancyRpartPlot((fit.pruned))


default_tree <- rpart(foodexpense~., data=train) #cp=0.01
summary(default_tree)
fancyRpartPlot(default_tree)


#Cross validation
V = 10 #V-fold CV
mse.train = 0; mse.test = 0
set.seed(0)
id = sample(1:V, nrow(train), replace = T)
for(i in 1:V) {
  print(i)
  ## Data partitioning
```

```
    val.index = which(id==i)
    mydata.train = train[-val.index,] #train data
    mydata.test   = train[val.index,] #test data
    ## Fitting
    tree.fit <- rpart(foodexpense~., data=mydata.train)
    ## Predicting and Evaluating
    yhat = predict(tree.fit, newdata=mydata.test)
    mse.test = mse.test + mean((mydata.test$foodexpense - yhat)^2)   # MSE
}

cv.mse.test_tree   = mse.test/V;   cv.mse.test_tree   # test CV MSE

#test MSE
yhat_tree = predict(default_tree, newdata=test)
test_mse_tree = mean((test$foodexpense-yhat_tree)^2)
test_mse_tree

##################################################
#######RandomForest
rf.foodexpense <- randomForest(foodexpense ~., data=train)
rf.foodexpense
# Predictions on the test data set
yhat.rf <- predict(rf.foodexpense, newdata = test)
plot(yhat.rf, test$foodexpense)
```

```
abline(0, 1, col='red')


test_mse_rf = mean((yhat.rf - test$foodexpense)^2)

test_mse_rf


#Cross validation

V = 10 #V-fold CV

mse.train = 0; mse.test = 0


set.seed(0)

id = sample(1:V, nrow(train), replace = T)

for(i in 1:V) {

  print(i)

  ## Data partitioning

  val.index = which(id==i)

  mydata.train = train[-val.index,] #train data

  mydata.test   = train[val.index,] #test data

  ## Fitting

  rf.foodexpense <- randomForest(foodexpense ~., data=mydata.train)

  ## Predicting and Evaluating

  yhat = predict(rf.foodexpense, newdata=mydata.test)

  mse.test = mse.test + mean((mydata.test$foodexpense - yhat)^2)   # MSE

}

cv.mse.test_rf   = mse.test/V;   cv.mse.test_rf   # test CV MSE
```

```
# variable importance plot
importance(rf.foodexpense)
varImpPlot(rf.foodexpense)


##############################################
### GAM (generalized additive model)
library(gam)
#validation_set approach로 validation mse 최소로 하는 df 찾기
set.seed(0)
index = sample(1:nrow(train), round(0.7*nrow(train)))
new_train = train[index,] #train data
new_test = train[-index,] #test data
error_list = c()
for (i in 2:5){
  for (j in 2:5){
    gam.fit <- lm(foodexpense~ns(Income, df=i)+ns(Age, df=j)+Education + Marital_Status
+ Customer_period + NumChild + NumResponded, data = new_train)
    pred <- predict(gam.fit, newdata=new_test)
    error <- mean((new_test$foodexpense-pred)^2)
    error_list <- append(error_list, error)
  }
}
error_list
which.min(error_list)
```

```
gam.fit <- lm(foodexpense ~ ns(Income, df = 4) + ns(Age, df = 3) + Education + Marital_Status + Customer_period + NumChild + NumResponded, data = train)

par(mfrow = c(2,4))

plot.Gam(gam.fit, se = TRUE, col = "red")

#Cross validation

V = 10 #V-fold CV

mse.train = 0; mse.test = 0



set.seed(0)

id = sample(1:V, nrow(train), replace = T)

for(i in 1:V) {

  print(i)

  ## Data partitioning

  val.index = which(id==i)

  mydata.train = train[-val.index,] #train data

  mydata.test   = train[val.index,] #test data

  ## Fitting

  gam.fit <- lm(foodexpense ~ ns(Income, df = 4) + ns(Age, df = 3) + Education + Marital_Status + Customer_period + NumChild + NumResponded, data = mydata.train)

  ## Predicting and Evaluating

  yhat = predict(gam.fit, newdata=mydata.test)

  mse.test = mse.test + mean((mydata.test$foodexpense - yhat)^2)   # MSE

}

cv.mse.test_gam   = mse.test/V;   cv.mse.test_gam   # test CV MSE
```

```
#test mse

gam.fit <- lm(foodexpense ~ ns(Income, df = 4) + ns(Age, df = 3) + Education + Marital_Status + Customer_period + NumChild + NumResponded, data = train)

yhat_gam = predict(gam.fit, newdata=test)

test_mse_gam = mean((test$foodexpense-yhat_gam)^2)

test_mse_gam


df <- data.frame('Linear Regression'=numeric(), 'Tree'=numeric(), 'Randomforest'=numeric(), 'GAM'=numeric())

df[1,] = c(sqrt(cv.mse.test_lm), sqrt(cv.mse.test_tree), sqrt(cv.mse.test_rf), sqrt(cv.mse.test_gam))

df[2,] = c(sqrt(test_mse_lm), sqrt(test_mse_tree), sqrt(test_mse_rf), sqrt(test_mse_gam))

rownames(df) = c('CV RMSE', "Test RMSE")

knitr::kable(df)
```

- **Codes for Classification (전처리 이후의 Python Code를 작성)**

```
x_train, x_test, y_train, y_test = train_test_split(df.drop('Response', axis=1), df['Response'], test_size=0.3, random_state=20211126)

print('Train: ', len(x_train))

print('Test: ',  len(x_test))

print('N/P Sample: ', Counter(y_train))

- 성능 평가를 위한 train/test set 분할


data = dict(Counter(y_train))

names = list(data.keys())

values = list(data.values())
```

```
plt.bar(range(len(data)), values, tick_label=names)

plt.show()
```

- train 자료의 약 84.47%가 'respose'에 1을 답하였다. 캠페인에 찬성한 의견에 대해서 학습하기 어려울 수 있음.

```
pd.set_option('display.max_columns', None)

x_train.columns
```

- 범주형 변수들 각각 1,0으로 선형 종속 자명.

```
scaler = MinMaxScaler()

# 교차검증시

scaler.fit(x_train)

x_train_scaled = scaler.transform(x_train)

x_test_scaled = scaler.transform(x_test)

x_train_scaled =pd.DataFrame(x_train_scaled)

x_test_scaled =pd.DataFrame(x_test_scaled)

x_train_scaled

sm1 = SMOTE(random_state=20211126, k_neighbors=3)

x_train_scaled, y_train = sm1.fit_resample(x_train_scaled, y_train.ravel())

print('Train: ', len(x_train_scaled))

print('Test: ',   len(x_test_scaled))

print('N/P Sample: ', Counter(y_train))


data = dict(Counter(y_train))

names = list(data.keys())
```

```
values = list(data.values())


plt.bar(range(len(data)), values, tick_label=names)

plt.show()
```

- SMOTE: KNN처럼 인근 데이터 포인트 잡아서 변동성 고려하여, 사이에 새로운 관측치들을 샘플링.(거리를 scaling 한 이유)

- 이외에도 다양한 방법들이 있음 => roc curve minimizing or stratified cross validation

```
x_train_scaled.columns = x_train.columns

x_test_scaled.columns = x_test.columns


# 3. Logistic Regression

## (1) all variables

log_clf1 =sm.Logit(y_train,x_train_scaled)

classifier1 = log_clf1.fit()

print(classifier1.summary2())

LR = LogisticRegression()

# K-Fold Validation

kfold = 10

# ACC Score

LR_cv_results_acc = cross_val_score(LR, x_train_scaled, y_train, cv=kfold, scoring='roc_auc')

msg = "%s k-fold AUC: %f (%f)" % ('LR', LR_cv_results_acc.mean(), LR_cv_results_acc.std())

print(msg)


## (2) variable selection
```

```
to_remove = ['Kidhome','Teenhome','Children', 'Family_Size','Income','MntSweetProducts',
'MntFruits', 'Dt_Customer_long', 'Complain', 'MntGoldProds']

x_logit = x_train_scaled.drop(to_remove, axis=1)

x_logit_test = x_test_scaled.drop(to_remove, axis=1)

log_clf2 =sm.Logit(y_train,x_logit)

classifier2 = log_clf2.fit()

print(classifier2.summary2())
```

- 로지스틱 모형의 장점: 해석

- 음의 영향 주요 변수: Recency, NumStorePurchases, Dt_Customer_recent

- 구매 후 일수가 오래 지날수록 캠페인 참여 줄어든다. Store에서 구매하는 횟수가 많을수록 캠페인 참여 줄어든다. 최근에 가입할수록 캠페인 응답 X

- 양의 영향 주요 변수: MntMeatProducts, NumCatalogPurchases, NumWebPurchases

- 육류소비가 많은 고객과 Catalog, Web에서 구매하는 횟수가 많을수록 캠페인 참여 오즈비 증가.

```
LR = LogisticRegression()

# K-Fold Validation

kfold = 10

# ACC Score

LR_cv_results_acc = cross_val_score(LR, x_logit, y_train, cv=kfold, scoring='roc_auc')

msg = "%s k-fold AUC: %f (%f)" % ('LR', LR_cv_results_acc.mean(), LR_cv_results_acc.std())

print(msg)
```

- backward selection을 사용한 logistic regression: 아주 약간 AUC 면적 감소

```
# 4. Adaboost

param_grid = {
```

```
        "learning_rate" : [0.1, 0.4, 0.7, 1, 1.3, 1.6],
        "base_estimator__max_depth" : [1, 5, 10, 15, 20, 25],
        "n_estimators": [100, 200, 300, 400, 500, 600]
                }
# Validation for Boosting Tree
clf = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(), random_state=20211126)
# run grid search
grid_search_clf = GridSearchCV(clf, param_grid=param_grid, scoring = 'roc_auc', cv=kfold)
grid_search_clf.fit(x_train_scaled,y_train)

print(grid_search_clf.best_params_)
print(grid_search_clf.best_score_)
param_grid = {
        "learning_rate" : [0.4],
        "base_estimator__max_depth" : [15],
        "n_estimators": [600, 800, 1000, 1200, 1400, 1600]
                }

# Validation for Boosting Tree
clf = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(), random_state=20211126)

# run grid search
grid_search_clf = GridSearchCV(clf, param_grid=param_grid, scoring = 'roc_auc', cv=kfold)
```

```
grid_search_clf.fit(x_train_scaled,y_train)


print(grid_search_clf.best_params_)

print(grid_search_clf.best_score_)

clf = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=15),
learning_rate=0.4, n_estimators=600, random_state=20211126)

clf.fit(x_train_scaled,y_train)


## 5. SVM

## (1) all variables

SVM=svm.SVC(kernel = 'rbf')

param_grid = {'gamma': [0.1, 0.5, 1, 2, 4],

                    'C' : [0.5, 1, 2.5, 5, 7.5, 10]}

grid = GridSearchCV(SVM, param_grid=param_grid,

                        scoring=['roc_auc'], refit='roc_auc',

                        return_train_score=True, cv=kfold)


grid.fit(x_train_scaled, y_train)

print(grid.best_params_)

print(grid.best_score_)

SVMmodel1=svm.SVC(kernel = 'rbf', gamma=2, C=7.5, random_state=20211126,
probability=True)


SVMmodel1.fit(x_train_scaled, y_train)
```

(2) variable selection1: using logistic regression coeff

- 비선형 서포트 벡터 머신에 피쳐 선택을 사용하는 것은 일반적으로 좋지 않은 아이디어라고 생각된다. 일반적으로 피쳐 선택을 위한 노력이 일반화 성능을 저하시킨다는 것. 따라서, logistic regression 결과에 의한 coefficient로 비교하고자 함.

```
SVM=svm.SVC(kernel = 'rbf')
param_grid = {'gamma': [0.5, 1, 2, 4, 6, 8],
                      'C' : [0.5, 1, 2.5, 5, 7.5, 10]}
grid = GridSearchCV(SVM, param_grid=param_grid,
                          scoring=['roc_auc'], refit='roc_auc',
                          return_train_score=True, cv=kfold)
grid.fit(x_logit, y_train)
print(grid.best_params_)
print(grid.best_score_)
SVMmodel2=svm.SVC(kernel = 'rbf', gamma=6, C=5, random_state=20211126, probability=True)
SVMmodel2.fit(x_logit, y_train)
```

```
# 6. test set performance
## (1) Logistic: all variables
pred1 = classifier1.predict(x_test_scaled)
roc_auc_score(y_test, pred1)
## (2) Logistic: variable selection
pred2 = classifier2.predict(x_logit_test)
roc_auc_score(y_test, pred2)
```

```
## (3) Adaboost

roc_auc_score(y_test, clf.predict(x_test_scaled))

## (4) SVM: all variables

roc_auc_score(y_test, SVMmodel1.predict(x_test_scaled))

## (5) SVM: variable selection

roc_auc_score(y_test, SVMmodel2.predict(x_logit_test))


# 7. ROC curve comparision

pred = classifier.predict(x_logit_test)

pred1 = clf.predict_proba(x_test_scaled)

pred2 = SVMmodel1.predict_proba(x_test_scaled)

fper, tper, thresholds = roc_curve(y_test, pred)

pred1 = pred1[:,1]

fper1, tper1, thresholds1 = roc_curve(y_test, pred1)

pred2 = pred2[:,1]

fper2, tper2, thresholds2 = roc_curve(y_test, pred2)


plt.plot(fper, tper, color='red', label='Logistic')

plt.plot(fper1, tper1, color='blue', label='Adaboost')

plt.plot(fper2, tper2, color='black', label='SVM')

plt.plot([0, 1], [0, 1], color='green', linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')
```

```
plt.title('Receiver Operating Characteristic Curve')

plt.legend()

plt.show()



# 8. Confusion Matrix

predictions1 = np.zeros(663)

predictions2 = np.zeros(663)

predictions3 = np.zeros(663)

predictions1[pred>0.5] = 1

predictions1[pred <= 0.5] = 0

predictions2[pred>0.7] = 1

predictions2[pred <= 0.7] = 0

predictions3[pred>0.9] = 1

predictions3[pred <= 0.9] = 0


print('Test ACC: ', accuracy_score(predictions1, y_test)) # cut_off:0.5

print(classification_report(y_test, predictions1))

print('Test ACC: ', accuracy_score(predictions2, y_test)) # cut_off:0.7

print(classification_report(y_test, predictions2))

print('Test ACC: ', accuracy_score(predictions3, y_test)) # cut_off:0.9

print(classification_report(y_test, predictions3))
```

- precision: 예측 한 것이 실제 맞는 경우.

- recall: 실제 target 값을 예측이 맞추는 경우.

- 원하는 목표에 맞춰서 cut-off value를 설정할 수 있다.

- 실제 1인 것들에 대해서 보수적으로 잘 맞춰주는 게 중요하다면, 0.5 (최대한 많은 고객으로부터 캠페인 이끌어내기)

- 1로 예측한 것들이 잘 맞추는 것이 중요하다면, 0.9 (비용절감)