Rakamin Academy

# Datalicious

## Dokumen Laporan Final Project

Marketing Campaign

Rakamin Academy

# Datalicious - Kelompok 1

**Members of Team :**

1. Nur Imam Masri
2. Astuti Rahmawati
3. Prasidya Bagaskara
4. Moh. Harwin Prayoga
5. Riskiyatul Hasanah
6. M Rayhan Azzindani
7. Siti Hajjah Mardiah
8. Christine
9. M. Ifzal Asril

# Git & Github

Link GDrive - Datalicious

Link Github - Datalicious

Rakamin Academy

# Stage 0

*Preparation*

# Latar Belakang Masalah

Rakamin Academy

| Problem Statement | | Roles |
|---|---|---|
| **Lotto Mart** adalah sebuah supermarket yang bergerak dibidang retail, menjual berbagai jenis produk seperti *Fish, Meat, Fruits, Sweet Products, Wines*, dan *Gold Products*. Selama 6 bulan terakhir, Marketing Team melakukan campaign berupa pemberian *discount vouchers* kepada semua customer melalui *Broadcast Message*. Namun, setelah *campaign* dilakukan, justru Lotto Mart menghadapi beberapa permasalahan sebagai berikut :<br>• **Response rate** dari marketing campaign yang dilakukan rendah yaitu sekitar 14.91%<br>• **Inefficient Cost** dalam melakukan marketing campaign<br>• **Profit tidak sebanding** dengan cost yang dikeluarkan | Berdasarkan hal tersebut, Marketing Team meminta tim data untuk menganalisis permasalahan yang terjadi. Selanjutnya perusahaan ingin membuat marketing campaign yang tepat sasaran sesuai dengan **karakteristik customer.** Strategi ini diharapkan mampu **meningkatkan response rate, meminimalisasi cost,** dan kemudian **meningkatkan profit.** | Sebagai **tim data** di Supermarket Lotto Mart, untuk menganalisis keberhasilan marketing campaign selanjutnya. Berikut adalah beberapa roles beserta PIC yang berkontribusi dalam menyelesaikan permasalahan di Lotto Mart:<br>• **Lead Data Science**<br>Sebagai koordinator project<br>PIC: **Nur Imam Masri**<br>• **Machine Learning Engineer**<br>Membuat model dan evaluasi Machine Learning<br>PIC: **Prasidya Bagaskara** dan **Moh. Harwin Prayoga**<br>• **Data Engineer**<br>Melakukan Data Preparation, Cleaning, dan Exploratory Data Analysis (EDA)<br>PIC: **M Rayhan Azzindani** dan **M. Ifzal Asril**<br>• **Business Analyst**<br>Membuat insight business<br>PIC: **Riskiyatul Hasanah** dan **Christine**<br>• **Data Analyst**<br>Membuat dashboard<br>PIC: **Siti Hajjah Mardiah** dan **Astuti Rahmawati** |

# Latar Belakang Masalah

| Goals | Objectives | Business Metrics |
|---|---|---|
| Perusahaan ingin meningkatkan **response rate** dan meminimalisasi **marketing campaign cost** sehingga dapat memaksimalkan **profit.** | • Membuat **classification model** untuk **memprediksi kelompok customer yang akan merespon campaign** agar dapat **meminimalisasi biaya pemasaran dan memaksimalkan keuntungan** pada campaign marketing berikutnya.<br>• Membuat **clustering model** untuk **memudahkan perusahaan menentukan target pelanggan yang tepat** | • **Response Rate/RR**<br>Persentase total customer response terhadap total delivered campaign .<br>***Indicator RR*** : 30% is good ( Efti 2018).<br>❖ **Net Profit Margin /NPM**<br>Mengukur net profit dibanding penjualannya. Semakin besar NPM, maka kinerja marketing campaign semakin efektif dan efisien (Handayani, Winarningsih 2020).<br>***Indicator NPM*** : 5% is low, 10-19% is average, 20% is good (Jayathilaka 2020).<br>❖ **Return of Investment /ROI**<br>CLV (customer lifetime value)) dibagi CAC((customer acquisition cost). Sebagai indikator kinerja perusahaan dan pembuatan keputusan para investor.<br>***Indicator ROI*** : 3:1 to 5:1 is good (Manzer 2017). |

# References


Efti S. 2018. *Sales Benchmarks: The 30/50 rule for cold emailing & cold calling*. diakses 20 Mei 2023. https://blog.close.com/sales-benchmarks/

Handayani N, Winarningsih S. 2020. The Effect of Net Profit Margin and Return on Equity Toward Profit Growth. *Moneter-Jurnal Akuntansi Dan Keuangan* 7(2): 198-204. https://doi.org/10.31294/moneter.v7i2.8701.

Jayathilaka AK. 2020. Operating Profit and Net Profit: Measurements of Profitability. *Open Access Library Journal* 7(12): 1-11. https://doi.org/10.4236/oalib.1107011.

Manzer D. 2017. *Five insights into measuring marketing ROI*. diakses 20 Mei 2023. https://growswyft.com/five-insights-to-measure-your-roi/.

Rakamin Academy

# Stage 1

*EDA, Insights & Visualization*

# Descriptive Analysis


## Marketing Campaign *Boost the profit of a marketing campaign*

- A response model can provide a significant boost to the efficiency of a marketing campaign by increasing responses or reducing expenses.
- The objective is to predict who will respond to an offer for a product or service

**Acknowledgements**
O. Parr-Rud. Business Analytics Using SAS Enterprise Guide and SAS Enterprise Miner. SAS Institute, 2014.

**Link Datasets :** https://www.kaggle.com/datasets/rodsaldanha/arketing-campaign

# Descriptive Analysis

## Dataset Description

The training dataset contains **2240 samples**. **29 features** and **1 target** boolean **variable "Response"** :

1. **Accepted/Responses Campaign**
   - **AcceptedCmp1** - 1 if customer accepted the offer in the 1st campaign, 0 otherwise
   - **AcceptedCmp2** - 1 if customer accepted the offer in the 2nd campaign, 0 otherwise
   - **AcceptedCmp3** - 1 if customer accepted the offer in the 3rd campaign, 0 otherwise
   - **AcceptedCmp4** - 1 if customer accepted the offer in the 4th campaign, 0 otherwise
   - **AcceptedCmp5** - 1 if customer accepted the offer in the 5th campaign, 0 otherwise
   - **Response (target)** - 1 if customer accepted the offer in the last campaign, 0 otherwise
   - **Complain** - 1 if customer complained in the previous 2 years
2. **Customer Information**
   - **ID - Customer's id**
   - **Year_Birth** - customer's year of birth
   - **Education** - customer's level of education
   - **Marital** - customer's marital status
   - **Kidhome** - number of small children in customer's household
   - **Teenhome** - number of teenagers in customer's household
   - **Income** - customer's yearly household income
   - **DtCustomer** - date of customer's enrolment with the company
   - **Recency** - number of days since the last purchase
3. **Sales Product Type**
   - **MntFishProducts** - amount spent on fish products in the last 2 years
   - **MntMeatProducts** - amount spent on meat products in the last 2 years
   - **MntFruits** - amount spent on fruits products in the last 2 years
   - **MntSweetProducts** - amount spent on sweet products in the last 2 years
   - **MntWines** - amount spent on wine products in the last 2 years
   - **MntGoldProds** - amount spent on gold products in the last 2 years
4. **Number of Purchases per Type**
   - **NumDealsPurchases** - number of purchases made with discount
   - **NumCatalogPurchases** - number of purchases made using catalogue
   - **NumStorePurchases** - number of purchases made directly in stores
   - **NumWebPurchases** - number of purchases made through company's web site
   - **NumWebVisitsMonth** - number of visits to company's web site in the last month
5. **Customer Information**
   - **Z_CostContact** = 3 (Cost to contact a customer)
   - **Z_Revenue** = 11 (Revenue after client accepting campaign)

# Descriptive Analysis

## Basic Datasets Information

- **Observations:**
  - Di dalam dataset terdiri dari **29 columns (28 features and 1 target boolean)** dan **2240 rows** data
  - Pada dataset terdapat 3 jenis tipe data yaitu : **int64, object, float64**
  - **Kolom Income** memiliki 2216 nilai non-null, dan **24 nilai null / missing values**

```
Shape of data : (2240, 29)
Number of rows : 2240
Number of columns : 29
```

```
 0   ID                   2240 non-null   int64
 1   Year_Birth           2240 non-null   int64
 2   Education            2240 non-null   object
 3   Marital_Status       2240 non-null   object
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64
 6   Teenhome             2240 non-null   int64
 7   Dt_Customer          2240 non-null   object
 8   Recency              2240 non-null   int64
 9   MntWines             2240 non-null   int64
 10  MntFruits            2240 non-null   int64
 11  MntMeatProducts      2240 non-null   int64
 12  MntFishProducts      2240 non-null   int64
 13  MntSweetProducts     2240 non-null   int64
 14  MntGoldProds         2240 non-null   int64
 15  NumDealsPurchases    2240 non-null   int64
 16  NumWebPurchases      2240 non-null   int64
 17  NumCatalogPurchases  2240 non-null   int64
 18  NumStorePurchases    2240 non-null   int64
 19  NumWebVisitsMonth    2240 non-null   int64
 20  AcceptedCmp3         2240 non-null   int64
 21  AcceptedCmp4         2240 non-null   int64
 22  AcceptedCmp5         2240 non-null   int64
 23  AcceptedCmp1         2240 non-null   int64
 24  AcceptedCmp2         2240 non-null   int64
 25  Complain             2240 non-null   int64
 26  Z_CostContact        2240 non-null   int64
 27  Z_Revenue            2240 non-null   int64
 28  Response             2240 non-null   int64
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
```

## Checking Duplicate Rows

- **Observations:** Data yang kita miliki tidak memiliki duplikat
- Ini dibuktikan pada **kolom ID** yang semua nilainya hanya muncul sekali

```
df[df.duplicated(keep=False)].sort_values(by=list(df.columns.values))
```

| ID | Year_Birth | Education | Marital_Status | Income | Kidhome | Teenhome | Dt_Customer | Recency |
|---|---|---|---|---|---|---|---|---|

```
df.duplicated().sum()
```

```
0
```

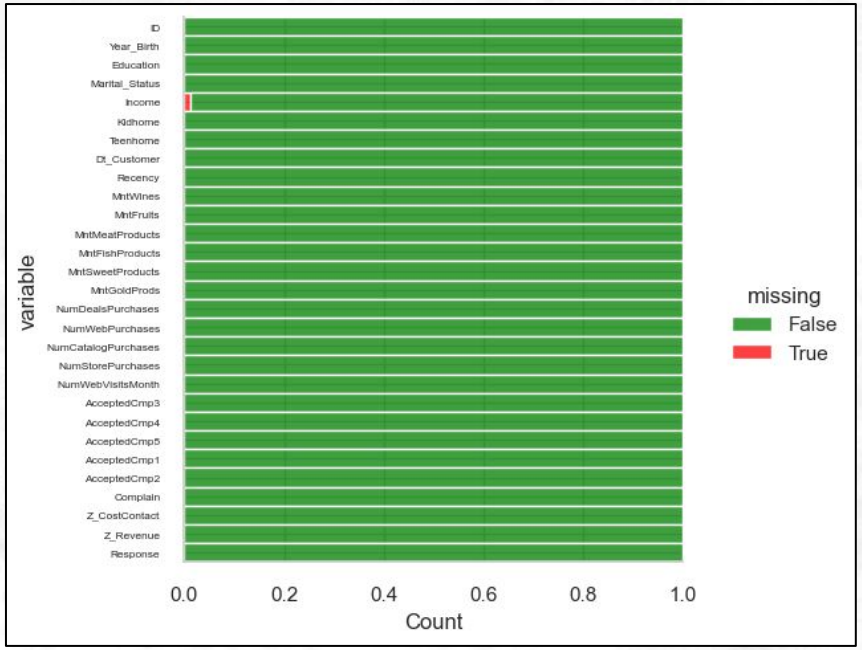
# Descriptive Analysis

## Checking Missing Values

- **Observations:**
    - Kolom **Income** memiliki **24 nilai null / missing values**, persentase sebesar **1.07% dari jumlah data**
- **Hal yang harus dilakukan saat Data Pre-Processing adalah:**
    - **Karena data yang dimiliki tidak terlalu banyak,** sehingga untuk Missing Values pada Income akan dilakukan **Imputation** pada tahap Data Preprocessingnya :
        - *Imputation (Median), karena Highly Positively Skewed*
        - *Multivariate Approach (MICE Imputation, KNN Imputer, dll)*

```
   #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              2240 non-null   int64
 1   Year_Birth      2240 non-null   int64
 2   Education       2240 non-null   object
 3   Marital_Status  2240 non-null   object
 4   Income          2216 non-null   float64
 5   Kidhome         2240 non-null   int64
 6   Teenhome        2240 non-null   int64
 7   Dt_Customer     2240 non-null   object
```

| | Total Null Values | Percentage | Data Type |
|---|---|---|---|
| Income | 24 | 1.071429 | int64 |
| ID | 0 | 0.000000 | int64 |
| Z_CostContact | 0 | 0.000000 | int64 |
| Complain | 0 | 0.000000 | object |
| AcceptedCmp2 | 0 | 0.000000 | object |
| AcceptedCmp1 | 0 | 0.000000 | float64 |
| AcceptedCmp5 | 0 | 0.000000 | int64 |
| AcceptedCmp4 | 0 | 0.000000 | int64 |
| AcceptedCmp3 | 0 | 0.000000 | object |



# Descriptive Analysis

## Data Types Information

**List of Column Types:**

- **Date**
  - *Dt_Customer*
- **Categorical (10 Columns) :**
  - **ID** → Nominal
  - **Education** → Ordinal (Levels : Basic - Graduation - 2n Cycle - Master - PhD)
  - **Marital_Status** → Nominal
  - **AcceptedCmp1, AcceptedCmp2, AcceptedCmp3, AcceptedCmp4, AcceptedCmp5, Complain, Response** → Nominal (Binary 0 & 1)

- **Continuous (18 Columns):**
  *Year_Birth, Income, Kidhome, Teenhome, Recency, MntWines, MntFruits, MntMeatProducts, MntFishProducts, MntSweetProducts, MntGoldProds, NumDealsPurchases, NumWebPurchases, NumCatalogPurchases, NumStorePurchases, NumWebVisitsMonth, Z_CostContact, Z_Revenue*

- **Di dalam dataset ditemukan tipe data yang kurang sesuai yaitu pada kolom berikut:**
  - **Dt-customer** : berbentuk **string/object,** tipe data kurang sesuai sehingga diubah menjadi **datetime (Memudahkan Extract)**
  - **Categorical (int64** dan **object)** diubah ke **category** untuk **Business Insight**
  - **Nama kolom dan isi sudah sesuai**

```
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2240 non-null   int64
 1   Year_Birth           2240 non-null   int64
 2   Education            2240 non-null   object
 3   Marital_Status       2240 non-null   object
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64
 6   Teenhome             2240 non-null   int64
 7   Dt_Customer          2240 non-null   object
 8   Recency              2240 non-null   int64
 9   MntWines             2240 non-null   int64
 10  MntFruits            2240 non-null   int64
 11  MntMeatProducts      2240 non-null   int64
 12  MntFishProducts      2240 non-null   int64
 13  MntSweetProducts     2240 non-null   int64
 14  MntGoldProds         2240 non-null   int64
 15  NumDealsPurchases    2240 non-null   int64
 16  NumWebPurchases      2240 non-null   int64
 17  NumCatalogPurchases  2240 non-null   int64
 18  NumStorePurchases    2240 non-null   int64
 19  NumWebVisitsMonth    2240 non-null   int64
 20  AcceptedCmp3         2240 non-null   int64
 21  AcceptedCmp4         2240 non-null   int64
 22  AcceptedCmp5         2240 non-null   int64
 23  AcceptedCmp1         2240 non-null   int64
 24  AcceptedCmp2         2240 non-null   int64
 25  Complain             2240 non-null   int64
 26  Z_CostContact        2240 non-null   int64
 27  Z_Revenue            2240 non-null   int64
 28  Response             2240 non-null   int64
dtypes: float64(1), int64(25), object(3)
```

# Descriptive Analysis

## Descriptive Numerical & Date Features Columns

Terdapat beberapa kolom yang memiliki nilai summary yang aneh diantaranya:

- **Year_Birth**
  - **Tahun kelahiran tertua (min)** yaitu **1893,** hal ini yang kemungkinan adanya salah input data sehingga data harus diproses lebih lanjut pada **Outlier**
  - Kemungkinan akan dilakukan **Feature extraction untuk mengambil data Umur/Age** pada range tahun saat ini 2014 (sesuai pada data)

| | Dt_Customer | Year_Birth | Income | Kidhome | Teenhome | Recency |
|---|---|---|---|---|---|---|
| count | 2240 | 2240.000000 | 2216.000000 | 2240.000000 | 2240.000000 | 2240.000000 |
| mean | 2013-07-10 10:01:42.857142784 | 1968.805804 | 52247.251354 | 0.444196 | 0.506250 | 49.109375 |
| std | - | 11.984069 | 25173.076661 | 0.538398 | 0.544538 | 28.962453 |
| min | 2012-07-30 00:00:00 | 1893.000000 | 1730.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 2013-01-16 00:00:00 | 1959.000000 | 35303.000000 | 0.000000 | 0.000000 | 24.000000 |
| 50% | 2013-07-08 12:00:00 | 1970.000000 | 51381.500000 | 0.000000 | 0.000000 | 49.000000 |
| 75% | 2013-12-30 06:00:00 | 1977.000000 | 68522.000000 | 1.000000 | 1.000000 | 74.000000 |
| max | 2014-06-29 00:00:00 | 1996.000000 | 666666.000000 | 2.000000 | 2.000000 | 99.000000 |

- **Income**
  - Memiliki nilai **Mean** yaitu sebesar **52247.25** dan **Median** sebesar **51381.5**, sehingga dapat disimpulkan bahwa data sedikit **Right-skewed Distribution** karena nilai **Mean > Median**
  - Mempunyai **Range 1730.0** (**min**) ke **666666.0** (**max**) yang sangat jauh, menandakan adanya **outliers.** sehingga perlu dilakukan **Log Transformation/Normalisasi atau Segmentasi** pada data income sebelum melanjutkan ke tahap pemodelan

- **Recency, Kidhome, Teenhome**
  - Memiliki nilai **mean** dan **median** yang sama artinya **kemungkinan** memiliki **normal-skewed distribution / bimodial** (akan **dicek pada univariate analysis**)
  - Untuk **Kidhome dan Teenhome** berpotensi untuk membuat **feature baru 'Dependents'** untuk lebih menggambarkan berapa jumlah anggota keluarga yg dependent

# Descriptive Analysis

## Descriptive Numerical & Date Features Columns

Terdapat beberapa kolom yang memiliki nilai summary yang aneh diantaranya:

- **Mount of Type Products**
  Pada beberapa kolom terdapat summary nilai yang memiliki **nilai Mean** dan **Median** memiliki rentang nilai terlalu jauh seperti pada kolom-kolom berikut :

  1. **MntWines**: memilik nilai **mean** sebesar **303.9** dan **median** sebesar **173.5**
  2. **MntFruits** : memilik nilai **mean** sebesar **26.3** dan **median** sebesar **8**
  3. **MntMeatProducts**: memilik nilai **mean** sebesar **166.9** dan **median** sebesar **67**
  4. **MntFishProducts** : memilik nilai **mean** sebesar **37.5** dan **median** sebesar **12**
  5. **MntSweetProducts** : memilik nilai **mean** sebesar **27.06** dan **median** **8**
  6. **MntGoldProds** : memilik nilai **mean** sebesar **44.02** dan **median** sebesar **24**

| | MntWines | MntFruits | MntMeatProducts | MntFishProducts | MntSweetProducts | MntGoldProds |
|---|---|---|---|---|---|---|
| count | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 |
| mean | 303.935714 | 26.302232 | 166.950000 | 37.525446 | 27.062946 | 44.021875 |
| std | 336.597393 | 39.773434 | 225.715373 | 54.628979 | 41.280498 | 52.167439 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 23.750000 | 1.000000 | 16.000000 | 3.000000 | 1.000000 | 9.000000 |
| 50% | 173.500000 | 8.000000 | 67.000000 | 12.000000 | 8.000000 | 24.000000 |
| 75% | 504.250000 | 33.000000 | 232.000000 | 50.000000 | 33.000000 | 56.000000 |
| max | 1493.000000 | 199.000000 | 1725.000000 | 259.000000 | 263.000000 | 362.000000 |

Jika dilihat dari beberapa nilai **mean** dan **median** yang memiliki **jarak agak aneh**, kemungkinan memiliki **jumlah outlier yang tinggi** dan **distribusi** yang **skewed.**

maka untuk Data Preprocessing perlu dilakukan **Log Transformation,** digunakan untuk mengubah data skewed mendekati / sesuai dengan normalitas.

# Descriptive Analysis

## Descriptive Numerical & Date Features Columns

Terdapat beberapa kolom yang memiliki nilai summary yang aneh diantaranya:

- **Moderately Positively Skewed** (Sedikit skew ke kanan) terdapat pada kolom berikut:
  1. **NumDealsPurchases**
  2. **NumWebPurchases**
  3. **NumCatalogPurchases**
  4. **NumStorePurchases**
  5. **NumWebVisitsMonth**

Jika dilihat pada data ini nilai **mean** dan **median** ditemukan memiliki jarak yang tidak lumayan jauh, namun **kemungkinan** memiliki **distribusi** yang **skewed** ke kanan, maka **perlu di visualisasi lebih lanjut,** kemudian untuk Data Preprocessing perlu dilakukan **Log Transformation**, digunakan untuk mengubah data skewed mendekati / sesuai dengan normalitas.

| | NumDealsPurchases | NumWebPurchases | NumCatalogPurchases | NumStorePurchases | NumWebVisitsMonth | Z_CostContact | Z_Revenue |
|---|---|---|---|---|---|---|---|
| count | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 | 2240.000000 |
| mean | 2.325000 | 4.084821 | 2.662054 | 5.790179 | 5.316518 | 3.000000 | 11.000000 |
| std | 1.932238 | 2.778714 | 2.923101 | 3.250958 | 2.426645 | 0.000000 | 0.000000 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 3.000000 | 11.000000 |
| 25% | 1.000000 | 2.000000 | 0.000000 | 3.000000 | 3.000000 | 3.000000 | 11.000000 |
| 50% | 2.000000 | 4.000000 | 2.000000 | 5.000000 | 6.000000 | 3.000000 | 11.000000 |
| 75% | 3.000000 | 6.000000 | 4.000000 | 8.000000 | 7.000000 | 3.000000 | 11.000000 |
| max | 15.000000 | 27.000000 | 28.000000 | 13.000000 | 20.000000 | 3.000000 | 11.000000 |

- **Z_CostContact, Z_Revenue**
  Dapat diketahui bahwa nilai **cost (3)** dan **revenue (11)** ini hanya memiliki satu nilai, sehingga akan di **Drop pada step modelling nantinya** karena tidak memberikan informasi yang signifikan terhadap model prediksi

# Descriptive Analysis

## Descriptive Categorical Features Columns

| | count | unique | top | freq |
|---|---|---|---|---|
| ID | 2240 | 2240 | 0 | 1 |
| Year_Birth | 2240 | 59 | 1976 | 89 |
| Education | 2240 | 5 | Graduation | 1127 |
| Marital_Status | 2240 | 8 | Married | 864 |
| AcceptedCmp1 | 2240 | 2 | 0 | 2096 |
| AcceptedCmp2 | 2240 | 2 | 0 | 2210 |
| AcceptedCmp3 | 2240 | 2 | 0 | 2077 |
| AcceptedCmp4 | 2240 | 2 | 0 | 2073 |
| AcceptedCmp5 | 2240 | 2 | 0 | 2077 |
| Complain | 2240 | 2 | 0 | 2219 |
| Response | 2240 | 2 | 0 | 1906 |

## Observation Result based on Descriptive Analysis

1. Terlalu banyak kategori pada kolom **ID**
2. Customer **banyak yang lahir** (**Year Birth**) pada tahun **1976** (**age = 38 years**) sebanyak **89 orang**
3. Kategori **Education**, **2n Cycle** dan **Master** memliki arti yang sama
4. Kategori **Education** pada customer yang mayoritas memiliki kategori pendidikan **Graduation** sebanyak **1127** orang, memiliki **nilai sangat besar** dibanding yang lain
5. Dalam kategori **Marital Status**,, customer **mayoritas sudah menikan (Married) 864 orang**
6. Dalam kategori **Marital Status**, **Single** dan **Alone** memiliki **arti yang sama**
7. Dalam kategori **Marital Status**, **Together** dan **Married** memiliki arti yang sama
8. Dalam kategori **Marital Status**, ada beberapa data yang tidak jelas apa yang dimaksud yaitu **Absurd** dan **YOLO** maka disarankan digabung dan diganti **Others**
9. Pada kategori **AcceptedCmp(1-5)**, customer mayoritas tidak merespon / accept dari campaign yang dilakukan
10. Pada kategori **Complain**, customer mayoritas tidak pernh complain dari campaign yang dilakukan
11. Target yang kita miliki terdapat pada kolom **Response** yang mana memiliki **ketimpangan yang sangat tinggi (Imbalanced Data)**

**- Tidak merespon = 1906**

**- Merespon = 334**

# Descriptive Analysis

## Descriptive Categorical Features Columns

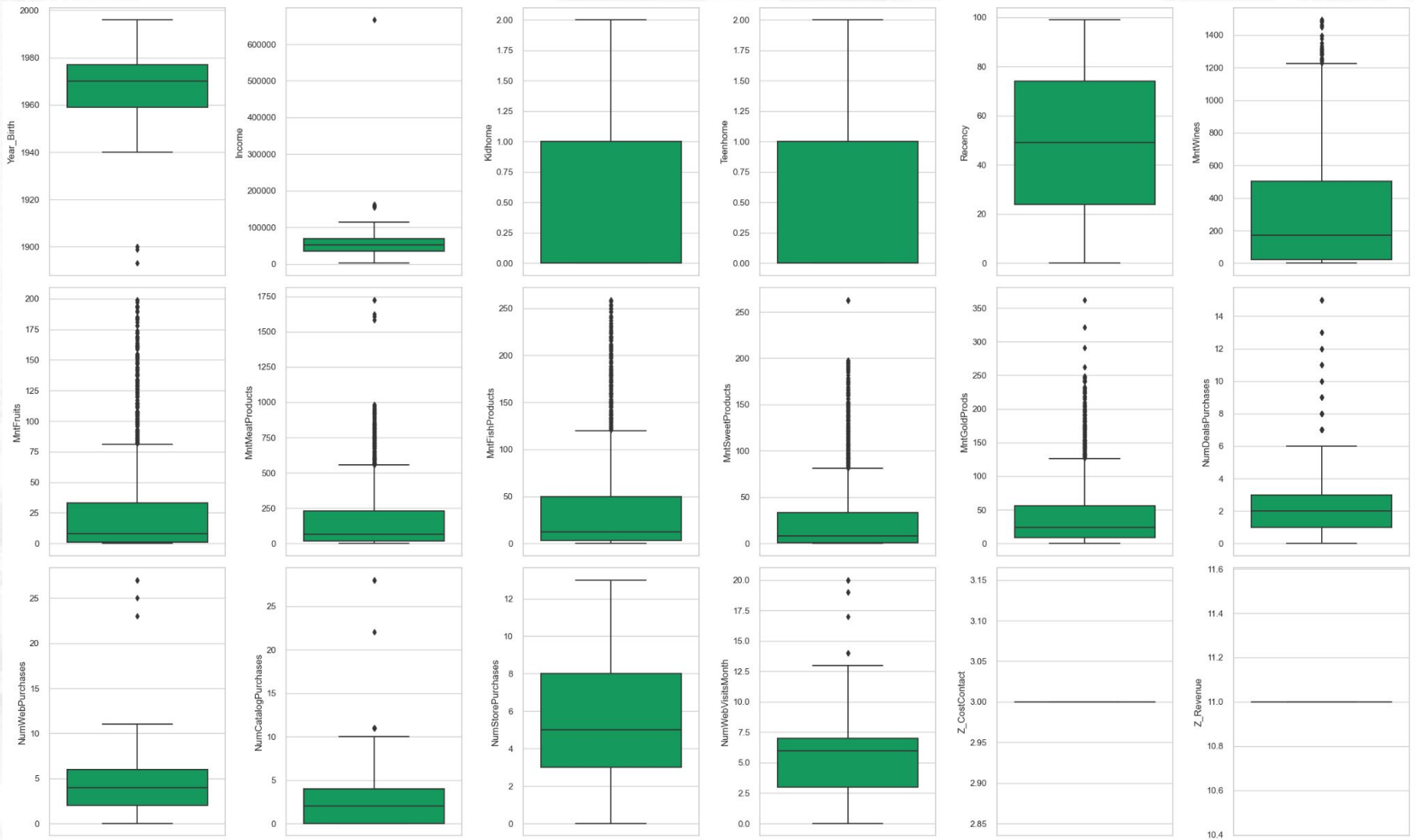
| | count | unique | top | freq |
|---|---|---|---|---|
| ID | 2240 | 2240 | 0 | 1 |
| Year_Birth | 2240 | 59 | 1976 | 89 |
| Education | 2240 | 5 | Graduation | 1127 |
| Marital_Status | 2240 | 8 | Married | 864 |
| AcceptedCmp1 | 2240 | 2 | 0 | 2096 |
| AcceptedCmp2 | 2240 | 2 | 0 | 2210 |
| AcceptedCmp3 | 2240 | 2 | 0 | 2077 |
| AcceptedCmp4 | 2240 | 2 | 0 | 2073 |
| AcceptedCmp5 | 2240 | 2 | 0 | 2077 |
| Complain | 2240 | 2 | 0 | 2219 |
| Response | 2240 | 2 | 0 | 1906 |

**Hal yang harus dilakukan saat Data Pre-Processing adalah:**

1. Akan dilakukan r**eplace data / menyatukan yang memiliki arti yang sama** agar mengurangi jumlah dimensi maupun **redudansi pada data**
2. Pada kolom **Response**, Sebaran kategori yang timpang pada target. Pada target, menyebabkan proses Machine Learning gagal. Oleh karena itu, perlu dilakukan **Sampling Data (Undersampling / Oversampling / Combinded / SMOTE / dll)**
3. Akan dilakukan **Feature Encoding** pada kolom **Education dan Marital_Status** untuk proses modelling, karena masih belum memiliki representasi nilai numerical

# Univariate Analysis

## DISTRIBUTION AND TABLE OF NUMERICAL VALUES



# Univariate Analysis

Terdapat ***outlier*** pada beberapa kolom di dalam dataset, diantaranya:

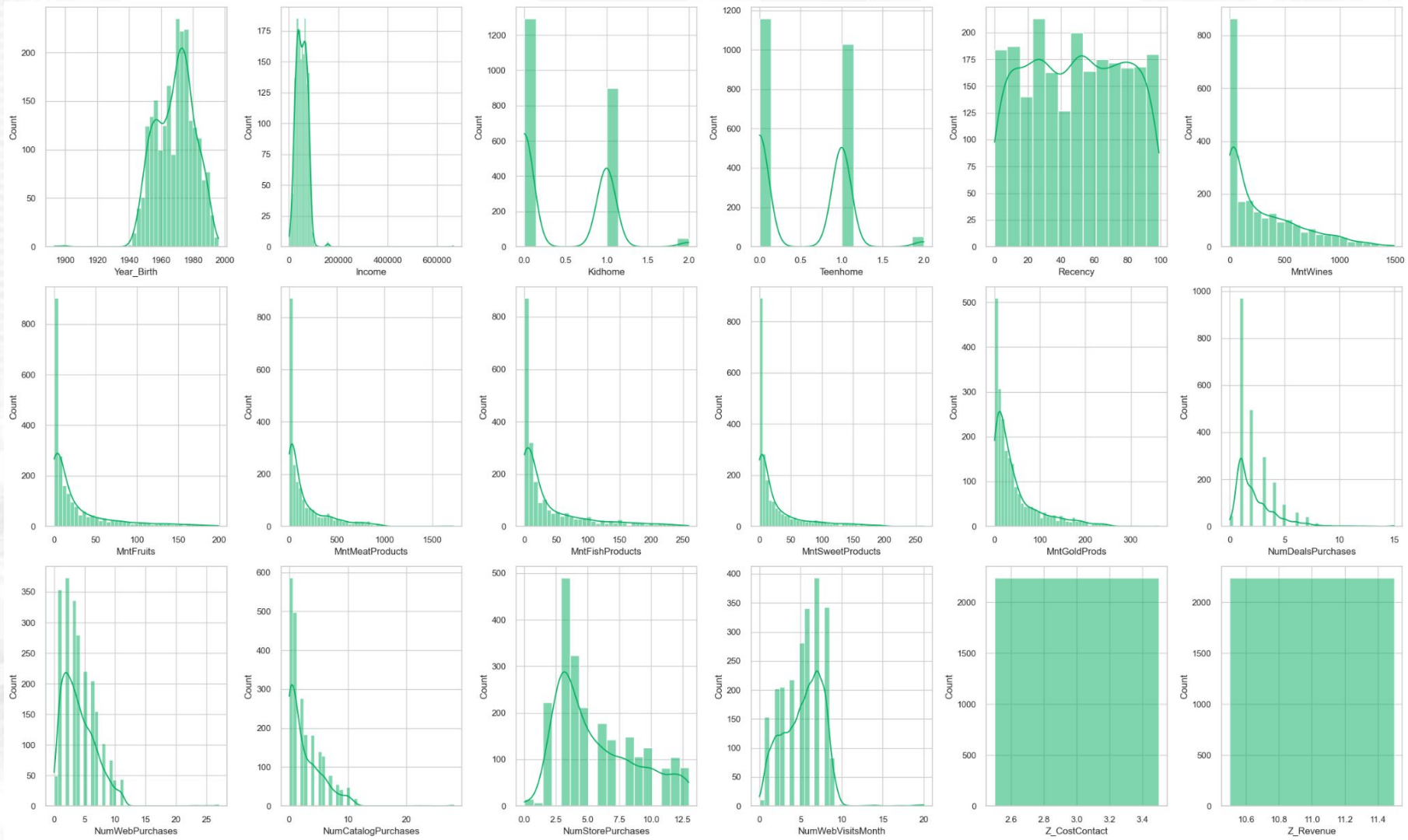
- ***Year_Birth***, ***outlier*** terjauh adalah **di bawah 1900**
- ***Income***, ***outlier*** terjauh adalah **di atas $600,000**
- ***MntWines***, ***outlier*** berada pada angka **1200 keatas**
- ***MntFruits***, ***outlier*** berada di **sekitar angka 80** sampai **200**
- ***MntMeatProducts***, ***outlier*** terjauh ada di **sekitar angka 1,750**
- ***MntFishProducts***, ***outlier*** berada di **sekitar angka 125** sampai **diatas 250**
- ***MntSweetProducts***, ***outlier*** terjauh berada di **sekitar angka 250**
- ***MntGoldProds***, ***outlier*** terjauh berada di **sekitar angka 350**
- ***NumDealsPurchases***, ***outlier*** terjauh berada di **angka 15**
- ***NumWebPurchases***, ***outlier*** berada di sekitar **angka 25**
- ***NumCatalogPurchases*, *outlier*** terjauh berada di atas **angka 25**
- ***NumWebVisitMonth***, ***outlier*** terjauh berada di **angka 20**

Hal yang harus dilakukan pada saat ***Data Pre-Processing*** adalah:

- Mengaplikasikan ***Log Transformation*** untuk ***Feature Scaling*** dan ***Handling Outlier*** yang mana transformasi ini ***de-emphasizes / minimize outliers*** dan dapat membantu untuk ***potentially obtain a bell-shaped / normal distribution***. Hal ini dilakukan karena **jumlah data yang terbatas yaitu sebanyak 2240 baris** data saja, sehingga menjadi pilihan terbaik karena dilakukan tanpa menghapus baris data.
- Alternatif lainnya, membersihkan data dengan cara **menghapus *outliers*** berdasarkan ***IQR*** atau ***Z-score***, akan tetapi hal ini akan mengurangi data yang dimiliki

# Univariate Analysis

## DISTRIBUTION AND TABLE OF NUMERICAL VALUES



# Univariate Analysis

## DISTRIBUTION AND TABLE

| | Column Name | Skewness | Kurtosis | Type of Distribution |
|---|---|---|---|---|
| 0 | Year_Birth | -0.350000 | 0.713000 | Moderately Normal Distribution (Symmetric) |
| 1 | Income | 6.759000 | 159.274000 | Highly Positively Skewed |
| 2 | Kidhome | 0.635000 | -0.781000 | Bimodal Distribution |
| 3 | Teenhome | 0.407000 | -0.987000 | Bimodal Distribution |
| 4 | Recency | -0.002000 | -1.202000 | Normal Distribution (Symmetric) |
| 5 | MntWines | 1.175000 | 0.595000 | Highly Positively Skewed |
| 6 | MntFruits | 2.101000 | 4.039000 | Highly Positively Skewed |
| 7 | MntMeatProducts | 2.082000 | 5.502000 | Highly Positively Skewed |
| 8 | MntFishProducts | 1.918000 | 3.087000 | Highly Positively Skewed |
| 9 | MntSweetProducts | 2.135000 | 4.364000 | Highly Positively Skewed |
| 10 | MntGoldProds | 1.885000 | 3.541000 | Highly Positively Skewed |
| 11 | NumDealsPurchases | 2.417000 | 8.914000 | Highly Positively Skewed |
| 12 | NumWebPurchases | 1.382000 | 5.688000 | Highly Positively Skewed |
| 13 | NumCatalogPurchases | 1.880000 | 8.027000 | Highly Positively Skewed |
| 14 | NumStorePurchases | 0.702000 | -0.623000 | Moderately Positively Skewed |
| 15 | NumWebVisitsMonth | 0.208000 | 1.815000 | Moderately Normal Distribution (Symmetric) |
| 16 | Z_CostContact | NaN | NaN | Uniform Distribution |
| 17 | Z_Revenue | NaN | NaN | Uniform Distribution |

Berdasarkan distribusi data di samping, dapat diketahui bahwa ada beberapa variabel yang memiliki *outlier* didalamnya dan beberapa memiliki *Skewed Distribution*. Berikut adalah beberapa variabel tersebut:

***A. Normal distribution***
- ***Recency*** *Normal Distribution (Symmetric)*
- ***Year_Birth*** *Moderately Normal Distribution (Symmetric)*
- ***NumWebVisitsMonth*** *Moderately Normal Distribution (Symmetric)*

***B. Uniform distribution***
- ***Z_CostContact*** *Uniform Distribution - Memiliki satu nilai saja*
- ***Z_Revenue*** *Uniform Distribution - Memiliki satu nilai saja*

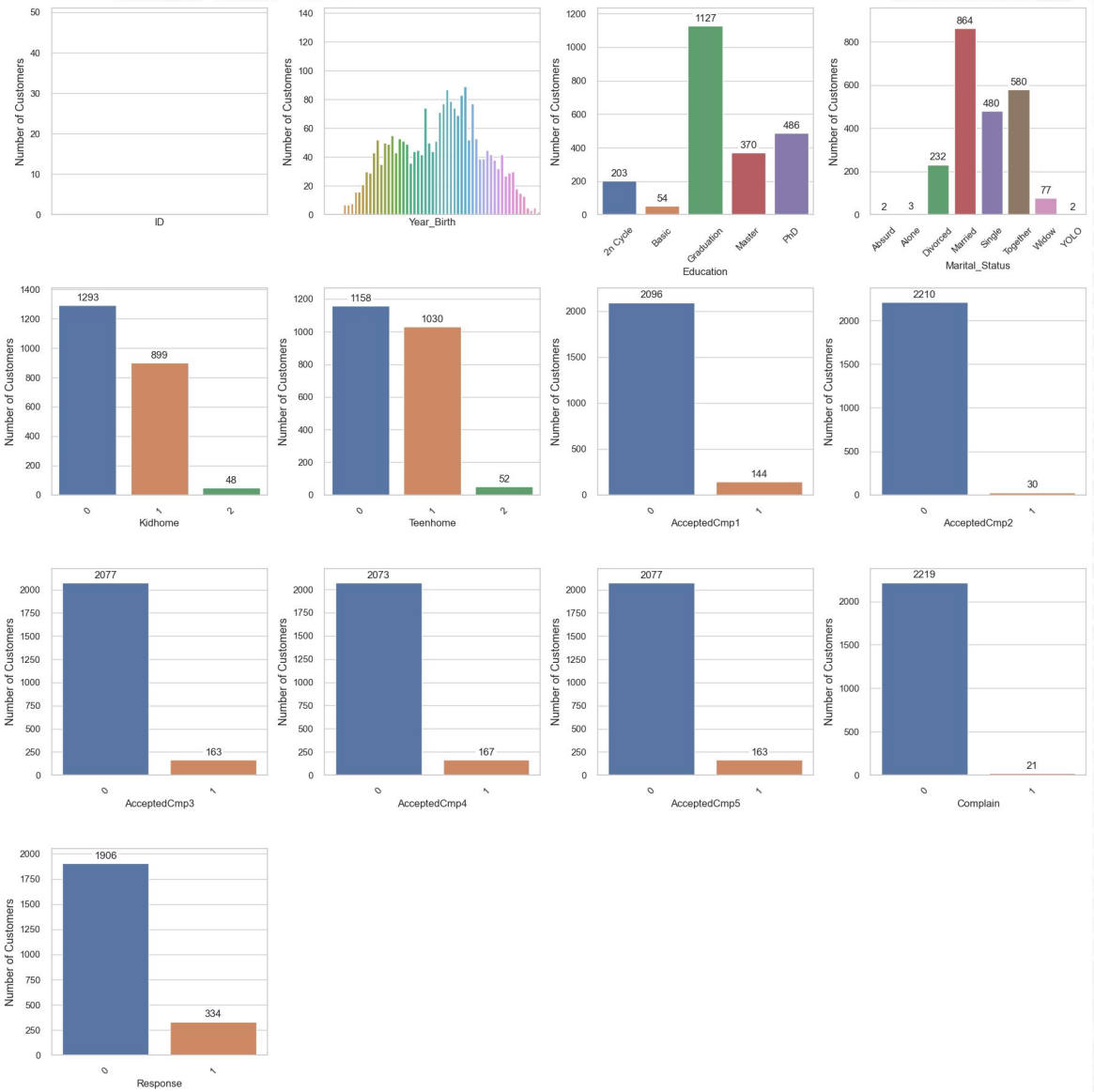
***C. Positive skewed distribution***
- ***Income*** *Income*
- *Amount of Wines Products* ***MntWines***
- *Amount of Fruits Products* ***MntFruits***
- *Amount of Meats Products* ***MntMeatProducts***
- *Amount of Fish Products* ***MntMeatProducts***
- *Amount of Sweet Products* ***MntSweetProducts***
- *Amount of Golds Products* ***MntGoldProds***
- *Number Deals Purchases* ***NumDealsPurchases***
- *Number Web Purchases* ***NumWebPurchases***
- *Number Catalog Purchases* ***NumCatalogPurchases***
- *Number Store Purchases* ***NumStorePurchases***

***D. Bimodal distribution***
- *Number of small children in customer's household* ***Kidhome***
- *Number of teenagers in customer's household* ***Teenhome***

**Rekomendasi pada data *pre-processing***:
Data yang mengalami ***Positive Skewed*** *Distribution* dilakukan ***Log Transformation*** sehingga data bisa menjadi ***normal*** *distribution*.

# Univariate Analysis

## Distribution of Categorical Values




1. Terlalu banyak kategori pada kolom **ID**
2. Kolom ***Education*** **dan** ***Marital_Status*** memiliki beberapa kategori **yang *value*-nya sama dan ambigu.**
   a. Kategori ***Education***, ***2n-Cycle*** dan ***Master*** memliki arti yang sama
   b. Kategori ***Education*** pada customer yang mayoritas memiliki kategori pendidikan ***Graduation*** sebanyak **1127** orang, memiliki **nilai sangat besar** dibanding yang lain
   c. Dalam kategori ***Marital Status***,, customer **mayoritas sudah menikan (*Married*) 864 orang**
   d. Dalam kategori ***Marital Status***, ***Single*** dan ***Alone*** memiliki **arti yang sama**
   e. Dalam kategori ***Marital Status***, ***Together*** dan ***Married*** memiliki arti yang sama
   f. Dalam kategori ***Marital Status***, ada beberapa data yang tidak jelas apa yang dimaksud yaitu ***Absurd*** dan ***YOLO*** maka disarankan digabung dan diganti ***Others***
3. Kolom ***Kidhome*** **dan** ***Teenhome*** mayoritas customer tidak memiliki anak dan remaja (*value* 0)
4. Kolom ***AcceptedCmp1***, ***AcceptedCmp2***, ***AcceptedCmp3*** ,***AcceptedCmp4***, ***AcceptedCmp5***, ***Complain***, **dan** ***Response*** *value* didominasi dengan value 0 (Tidak *Response* / *Complain*)
5. Target yang kita miliki terdapat pada kolom ***Response*** yang mana memiliki **ketimpangan yang sangat tinggi (*Imbalanced Data*)**
   **- Tidak merespon = 1906**
   **- Merespon = 334**

# Univariate Analysis

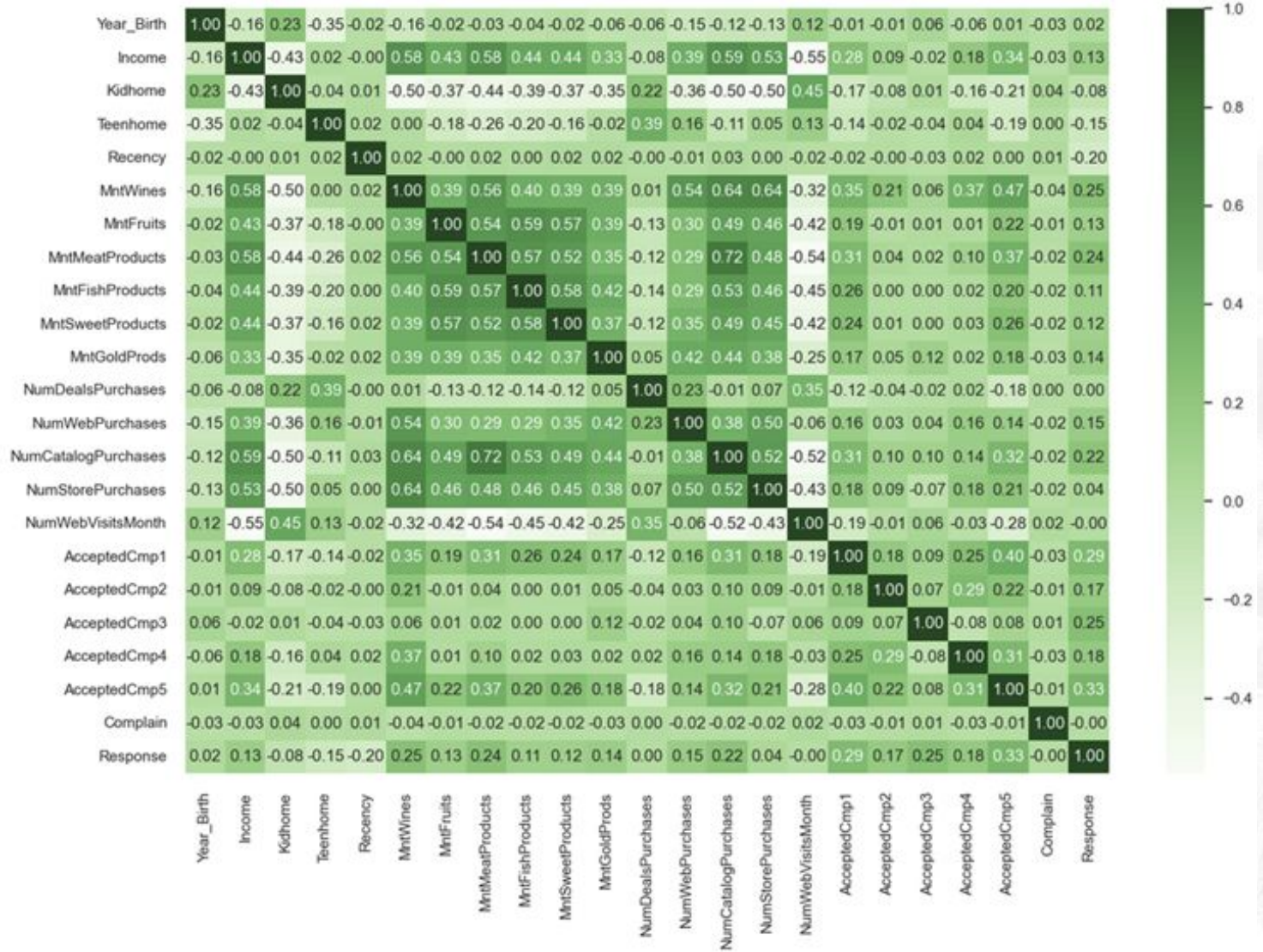
## Note for Data Pre-Processing Stage

**Hal yang harus dilakukan saat *Data Pre-Processing*:**

1. Kolom ***ID*** di drop untuk proses modelling
2. Dari kolom ***Year_Birth*** dibuat kolom baru yaitu kolom ***Age*** yang menunjukkan umur seorang customer.
3. Akan dilakukan ***replace data* / menyatukan yang memiliki arti yang sama** agar mengurangi jumlah dimensi maupun redudansi pada data
4. Melakukan ***Label Encoding*** pada kolom ***Education*.**
5. Melakukan ***One Hot Encoding* (*OHE*)** pada kolom ***Marital_Status*.**
6. Pada kolom ***Response*,** Sebaran kategori yang timpang pada *feature* mengindikasikan ketidakgunaan *feature*. Pada *target*, menyebabkan proses *Machine Learning gagal*. Oleh karena itu, perlu dilakukan ***Sampling Data* (*Undersampling* / *Oversampling* / *Combinded* / *SMOTE* / dll)**

# Multivariate Analysis

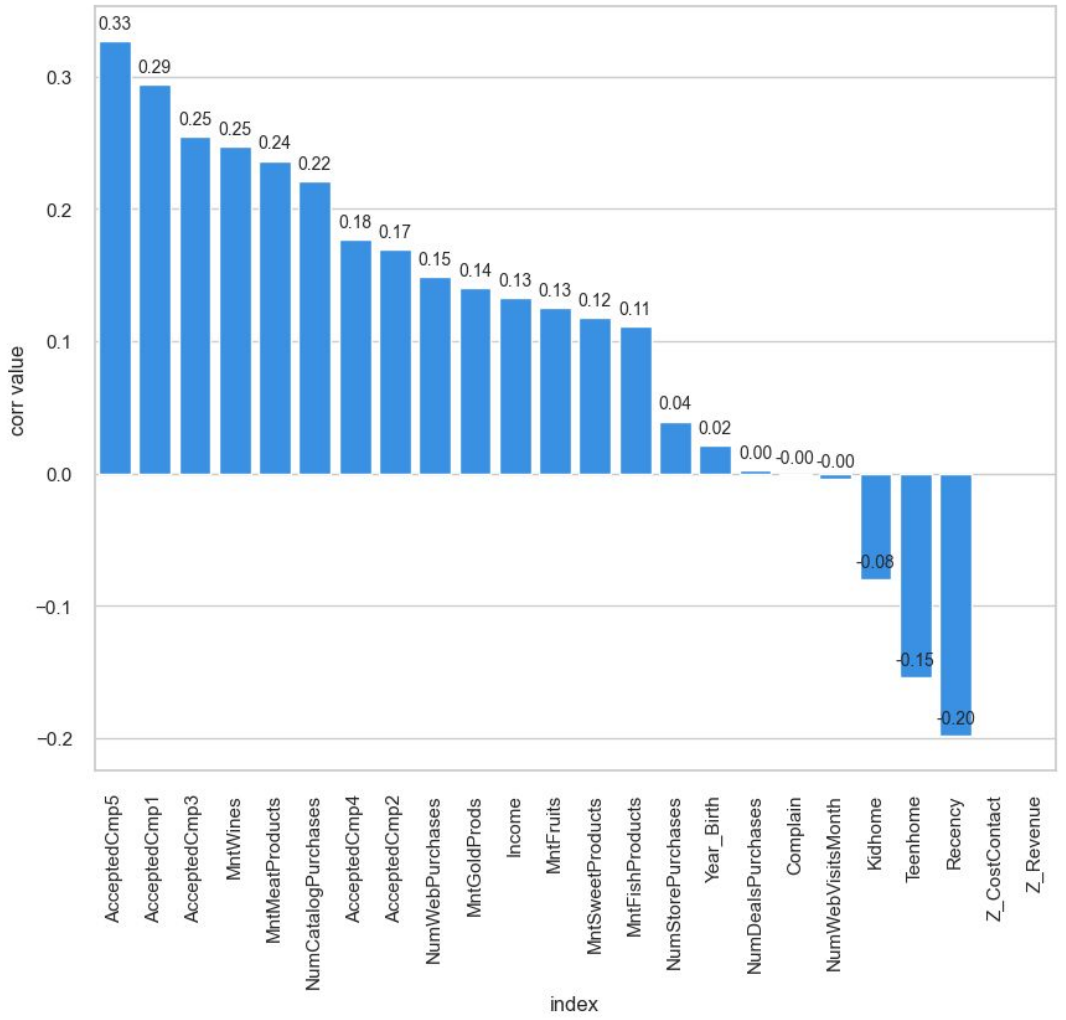
## HEATMAP AND TABLE




| | index | corr value | Corr Type |
|---|---|---|---|
| 0 | Response | 1.000000 | Positif |
| 1 | AcceptedCmp5 | 0.326634 | Positif |
| 2 | AcceptedCmp1 | 0.293982 | Positif |
| 3 | AcceptedCmp3 | 0.254258 | Positif |
| 4 | MntWines | 0.247254 | Positif |
| 5 | MntMeatProducts | 0.236335 | Positif |
| 6 | NumCatalogPurchases | 0.220810 | Positif |
| 7 | Recency | 0.198437 | Negatif |
| 8 | AcceptedCmp4 | 0.177019 | Positif |
| 9 | AcceptedCmp2 | 0.169293 | Positif |
| 10 | Teenhome | 0.154446 | Negatif |
| 11 | NumWebPurchases | 0.148730 | Positif |
| 12 | MntGoldProds | 0.139850 | Positif |
| 13 | Income | 0.133047 | Positif |
| 14 | MntFruits | 0.125289 | Positif |
| 15 | MntSweetProducts | 0.117372 | Positif |
| 16 | MntFishProducts | 0.111331 | Positif |
| 17 | Kidhome | 0.080008 | Negatif |
| 18 | NumStorePurchases | 0.039363 | Positif |
| 19 | Year_Birth | 0.021325 | Positif |
| 20 | NumWebVisitsMonth | 0.003987 | Negatif |
| 21 | NumDealsPurchases | 0.002238 | Positif |
| 22 | Complain | 0.001707 | Negatif |

# Multivariate Analysis

## HEATMAP AND TABLE




**Korelasi Feature dengan Response :**

A. Top 10 Yang berkolerasi tinggi ke target sebagai berikut, Kemungkinan besar top ini bisa menjadi feature yang paling relevan dan harus dipertahankan:
   - **AcceptedCmp5** - 0.32 – Positif
   - **AcceptedCmp1** - 0.29 – Positif
   - **AcceptedCmp3** - 0.25 – Positif
   - **MntWines** - 0.24 – Positif
   - **MntMeatProducts** - 0.23 – Positif
   - **NumCatalogPurchases** - 0.22 – Positif
   - **Recency** - 0.19 – Negatif
   - **AcceptedCmp4** - 0.17 – Positif
   - **AcceptedCmp2** - 0.16 – Positif
   - **Teenhome** - 0.15 – Negatif

B. Korelasi kolom **Response** dengan kolom lainnya cenderung **rendah**. Dari seluruh korelasi antara feature-target berada di range **0.00** sampai **0.33**. Oleh karena itu, kami memutuskan untuk membuat nilai threshold di angka **0.15**. Feature-feature di atas yang kemungkinan kami pertahankan adalah feature yang memiliki nilai korelasi **>0.15**.

# Multivariate Analysis

**Selain itu, pada korelasi antar-feature, terdapat pola yang menarik sebagai berikut:**

**A.** Untuk kolom **Complain**, **Z_CostContact** dan **Z_Revenue** berpotensi untuk dihapus, karena tidak memiliki korelasi dibanding kolom lainnya

**B.** **Year Birth**
- **Year_Birth** berkolerasi positif dengan **Kidhome**, bisa dikatakan semakin muda maka semakin banyak pula anak kecilnya sedangkan untuk kolom **Teenhome** berkolerasi negatif atau semakin tua customer maka jumlah anak remaja semakin banyak.

**C.** **Kidhome & Teenhome**
- Customer yang (**Kidhome**) memiliki anak kecil **Income** nya cenderung rendah.
- Customer yang (**Kidhome**) memiliki anak kecil cenderung lebih sering mengunjungi **web** dan melakukan pembelian ketika sedang diskon.
- Customer yang (**Teenhome**) memiliki anak remaja cenderung lebih banyak melakukan pembelian ketika sedang **diskon**.

**D.** **Income**
- Customer yang memiliki **Income** yang tinggi cenderung banyak melakukan pembelian.
- Kolom **Income** berkorelasi **positif** cukup besar dengan **MnWines**, **MntFruits**, **MntMeatProduct**, **MntFishProduct**, **MntSweetProduct** dan **MntGoldProduct**.
- Kolom **Income** berkorelasi **positif** cukup besar dengan **NumWebPurchases**, **NumCatalogPurchases**, dan **NumStorePurchases** sedangkan dengan kolom **NumWebVisitsPurchases** berkorelasi **negatif** cukup besar, bisa dikatakan bahwa semakin besar **income** customer maka mayoritas tempat yang dipilih untuk melakukan pembelian adalah **Web**, **Catalog** dan **Store**, sedangkan customer yang memiliki **income** rendah cenderung lebih banyak melakukan pembelian melalui **web** atau lebih sering mengunjungi web.

# Multivariate Analysis

**Lanjutan korelasi antar-feature :**

**D. Product**

- Kolom **MntWines** berkorelasi **positif** cukup besar dengan **MntMeatProduct**, kemungkinan customer membeli wine juga membeli meat.
- Kolom **MntFruits** berkorelasi **positif** cukup besar dengan **MntMeatProduct**, **MntFishProduct** dan **MntSweetProduct**, kemungkinan ketika customer membeli product tersebut bersamaan.
- **MntMeatProduct** berkorelasi positif cukup besar dengan **NumCatalogPurchases**, bisa dikatakan sebagian besar pembelian meat product dilakukan melalui catalog.
- **MntWines** paling banyak dibeli melalui **katalog** dari pada metode pembelian lainnya.

**E. Purchases**

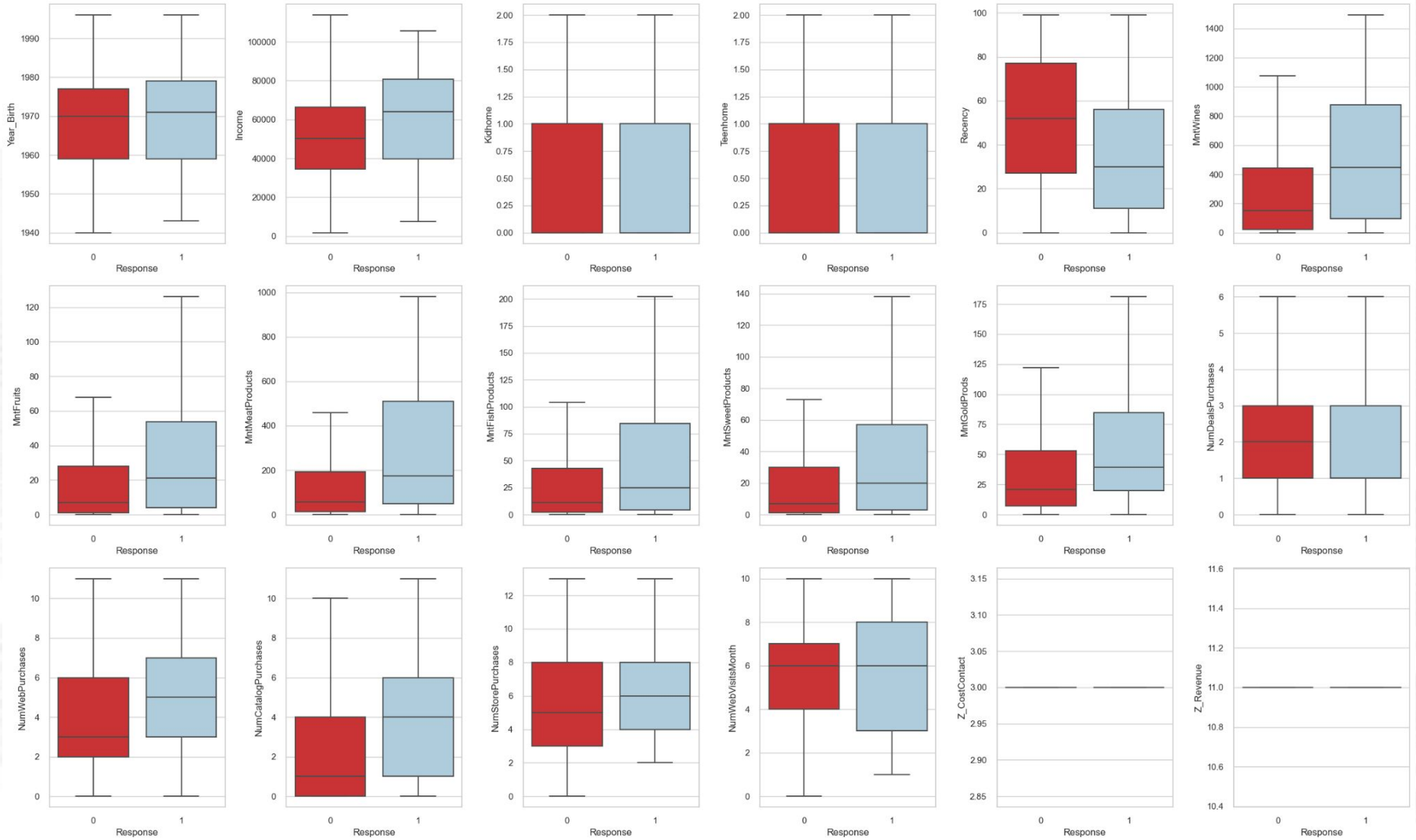
- Kolom **NumDealsPurchases** berkorelasi **positif** dengan **NumWebVisitMonth**, ketika sedang diskon customer yang mengunjungi web meningkat.
- Kolom **NumWebVisitMonth** berkorelasi **negatif** cukup besar dengan **NumCatalogPurchases** dan **NumStorePurchases**, ketika customer lebih sering mengunjungi web maka pembelian melalui catalog dan store menurun.

**F. Additional**

- Kombinasi 5 Kolom product cukup tinggi nilai korelasinya. Oleh karena itu, customer cenderung suka membeli lebih dari 1 product dalam sekali berbelanja.
- Produk **Wines** dan **Gold** lebih banyak dibeli menggunakan **website**. Sedangkan **Fruits**, **Meat**, **Fish** dan **Sweet** dibeli melalui **Store** maupun **Catalog**.
- Produk yang ditawarkan menggunakan **Deals** (potongan harga) belum terlalu menarik minta customer karena korelasi dengan kolom produk apapun sangat **rendah**.
- Customer yang menggunakan **Deals** lebih banyak customer yang menggunakan **Website** untuk membeli barang/produk.

# Multivariate Analysis

## BOXPLOT VS RESPON

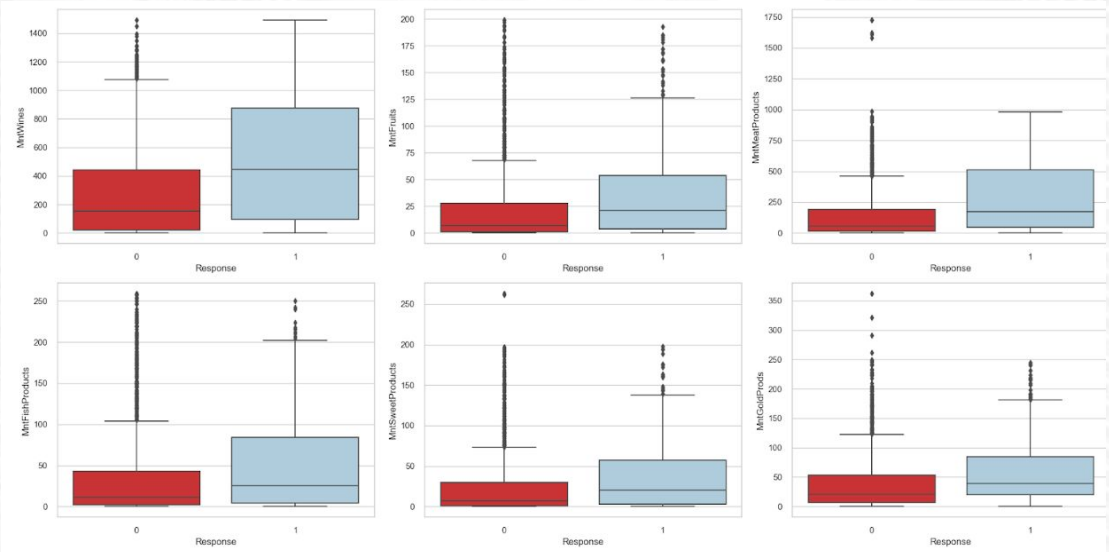



- **Income**, Customer yang merespon campaign cenderung memiliki income yang lebih tinggi dengan rata-rata income yang dimiliki customer sekitar 65000, dibanding customer yang tidak merespon campaign memiliki income rata-rata hanya sekitar 50000.
- **Recency,** Customer yang merespon campaign cenderung lebih aktif untuk berbelanja dengan rata-rata hari terakhir pembelian produk sekitar 30 hari, dibanding customer yang tidak merespon campaign memiliki rata-rata hari terakhir pembelian produk sekitar lebih dari 50 hari

# Multivariate Analysis

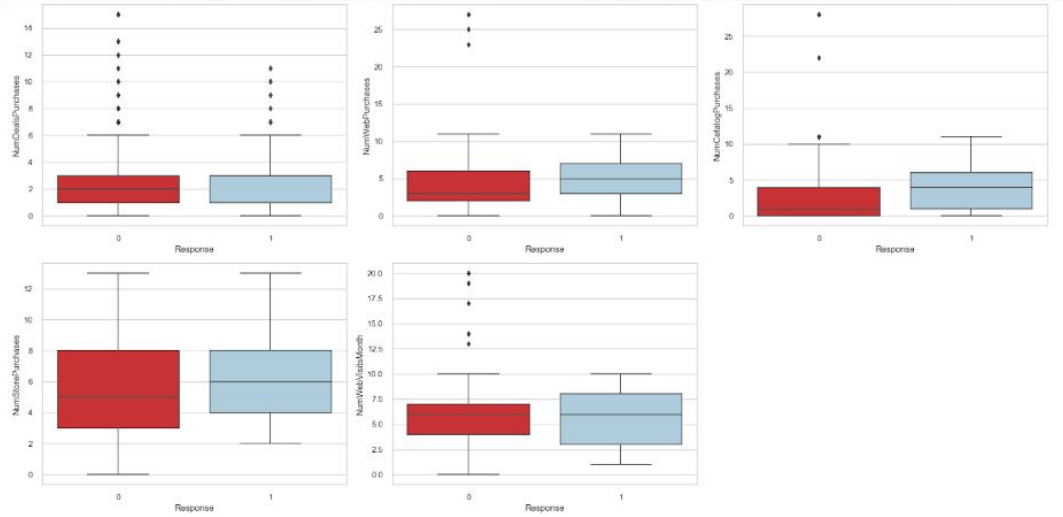
## SIMPLIFIED BOXPLOT VS RESPON

Tim melakukan '**Simplified Analysis based on MntProducts**. Pada dataset ini terdapat banyak jenis produk yang ditawarkan seperti **MntWines, MntFruits, MntMeatProducts, MntFishProducts, MntSweetProducts, MntGoldProducts**.




Berdasarkan boxplot diatas dapat disimpulkan bahwa, pada **Response yang menerima (values 1)**, memiliki nilai **Mount** (Jumlah Pembelian) di tiap product **lebih tinggi** daripada yang tidak merespon (values 0)

Adapun '**Simplified Analysis based on Type Purchases**' yang terdiri dari **NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth'**



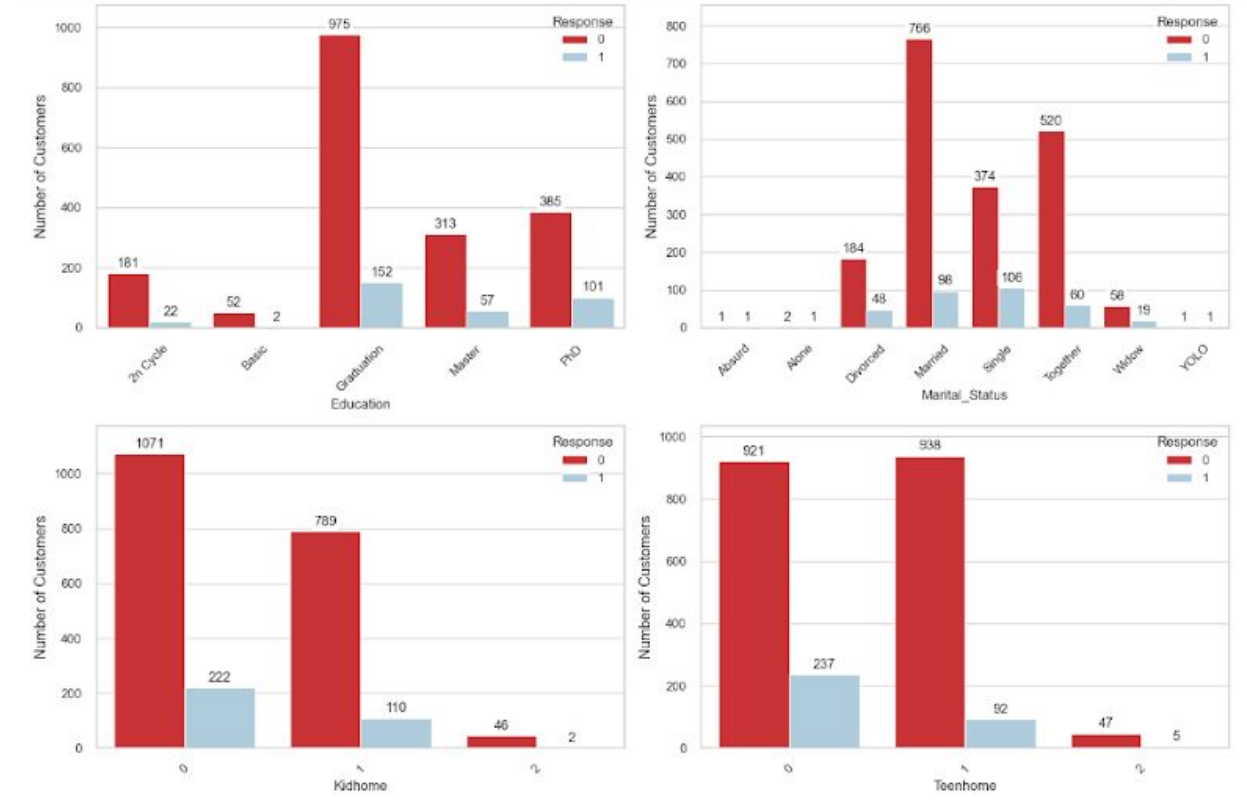

1. Untuk pembelian melalui **Web, Catalog, Web**, Pada **Response yang menerima (values 1)**, memiliki nilai **Purchases** (Jumlah Pembelian) lumayan sedikit **lebih tinggi** daripada yang tidak merespon (values 0)
2. Pada pembelian melalui **Deals / Discount** dan **Store** memiliki nilai yang **tidak terlalu berbeda**

# Multivariate Analysis

## SIMPLIFIED COUNTPLOT VS RESPON

Pada bagian ini, Tim melakukan analisis berdasarkan **Status Customer dan Campaign/Complain terhadap Response.**

***'Simplified based on Status Customer'***



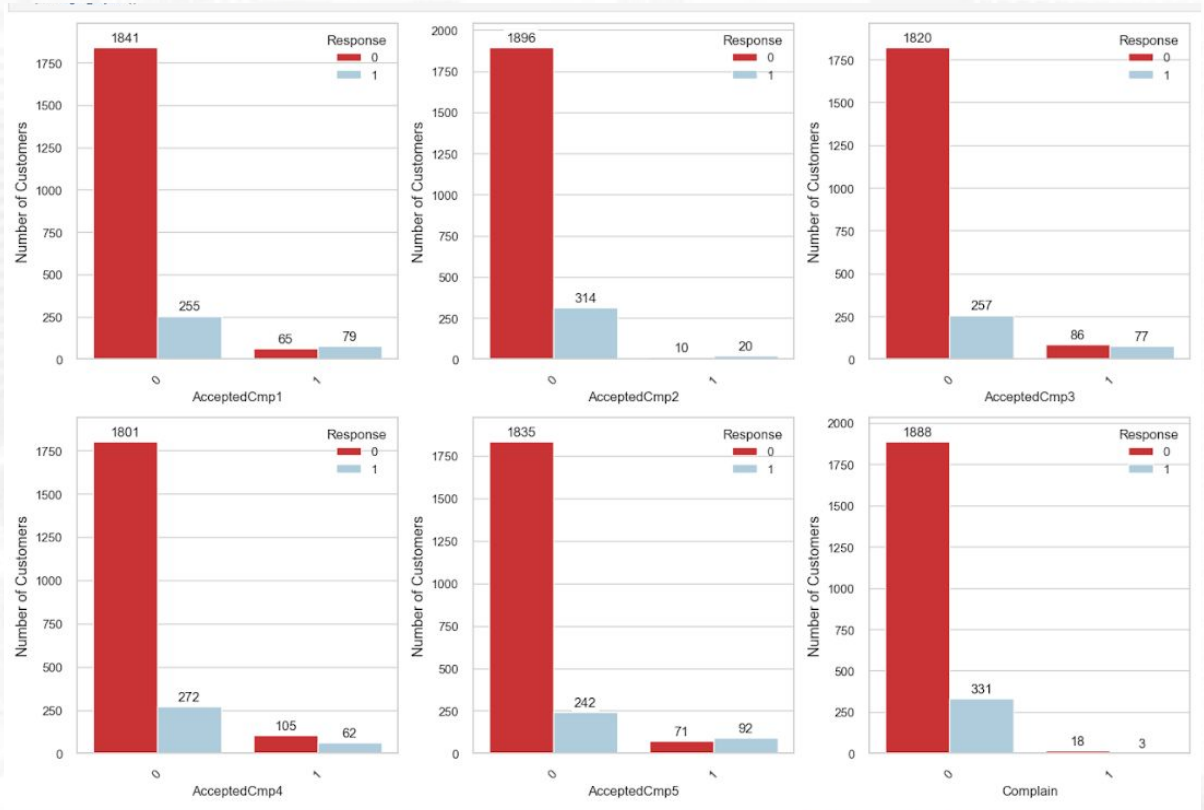

Berdasarkan countplot disamping, dapat disimpulkan bahwa:

1. **Education**
   **Graduation, PhD dan Master** memiliki jumlah respon yang tinggi, masing-masing ada pada ratio perbandingan respon > 13.4% (maks 20%)
2. **Marital_Status**
   **Single, Married, Together, dan Divorced** memiliki jumlah respon yang tinggi, namun ratio perbandingan respon vs no responnya < 22.4% (maks 50%)
3. **Kidhome & Teenhome**
   Semakin tinggi jumlah anak/remaja yang dimiliki customer, maka semakin kecil kemungkinan customer menerima Response (marketing campaign terakhir), sehingga lebih baik perusahaan menargetkan campaign kepada customer yang tidak memiliki anak/remaja. Begitupun pada ratio perbandingan respon vs no responnya menurun.

# Multivariate Analysis

## SIMPLIFIED BOXPLOT VS RESPON

***'Simplified based on Campaign/Complain'***



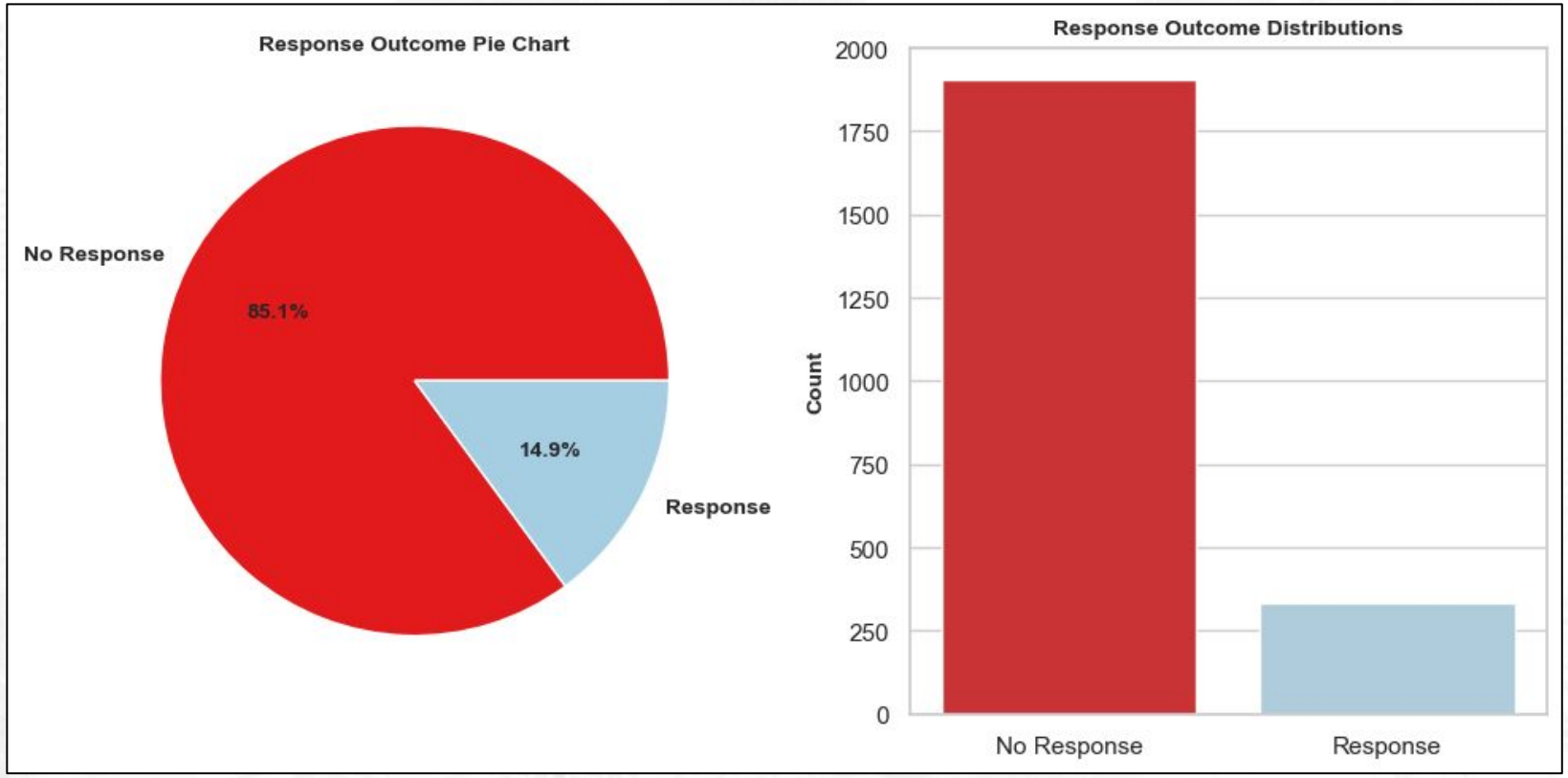

Berdasarkan countplot disamping dapat disimpulkan bahwa:

1. **AcceptedCmp columns**
   - Dari segi Response tertinggi cenderung pada yang tidak accept campaign.
   - Dari yang Accept Campaign, hanya pada 1, 3, 4, 5 yang memiliki nilai yang kebih besar. Bisa juga dilihat pada ratio perbandingan respon vs no responnya 33-56%, berarti tidak berbeda signifikan dan perbandingannya lumayan sama.
   - Pada Acccept Campaign 2 Jumlahnya sangat sedikit, namun ratio perbandingan respon vs no responnya yang paling tinggu 66%.
2. **Complain**
   - Dari segi Response tertinggi cenderung pada yang tidak ada complain pada campaign

# Business Insight

## Response Ratio

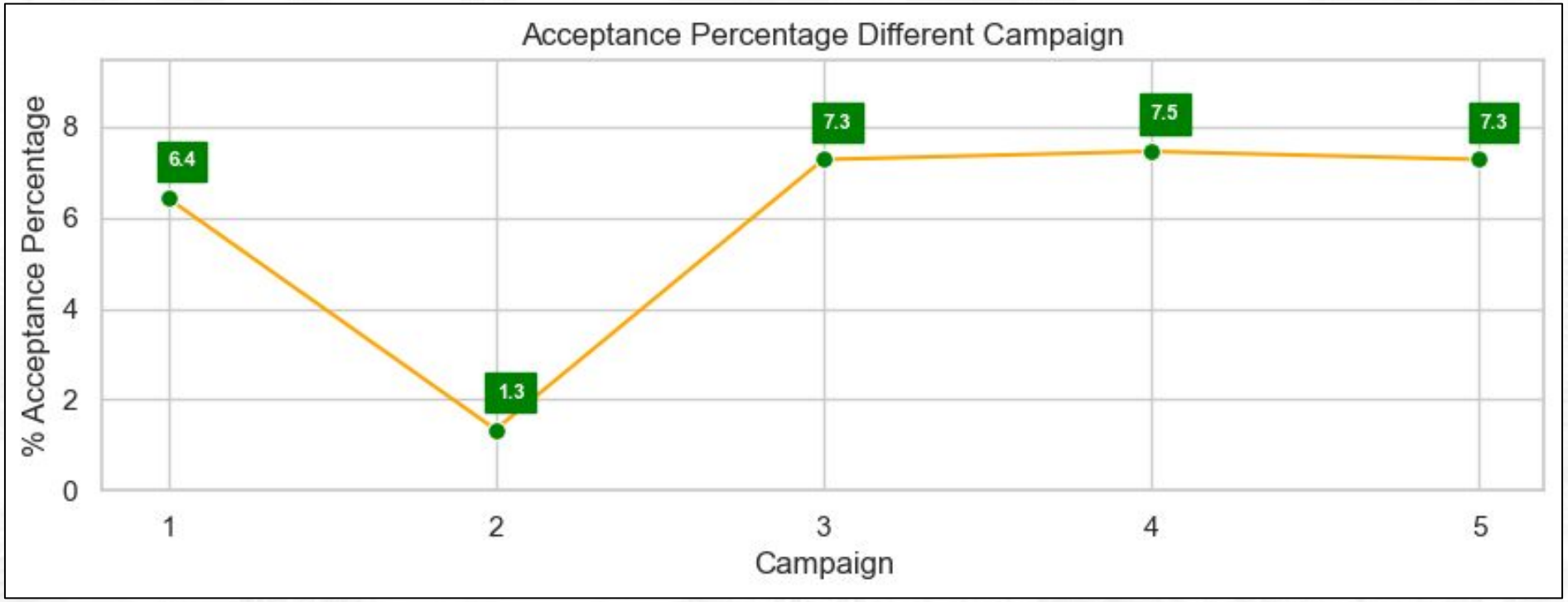
| Response | Total | % |
|---|---|---|
| No | 1906 | 85.090000 |
| Yes | 334 | 14.910000 |




- Jumlah yang **Response** signifikan lebih kecil dibandingkan yang Tidak Merespon **No Respon**, dengan ratio **14.9%**
- Ini berarti adanya **data imbalance** pada campaign terakhir (ke-6) perusahaan
- Sehingga bisa dilakukan **upaya peningkatan proses marketing campaign** agar lebih banyak customer yang merespon campaign

# Business Insight

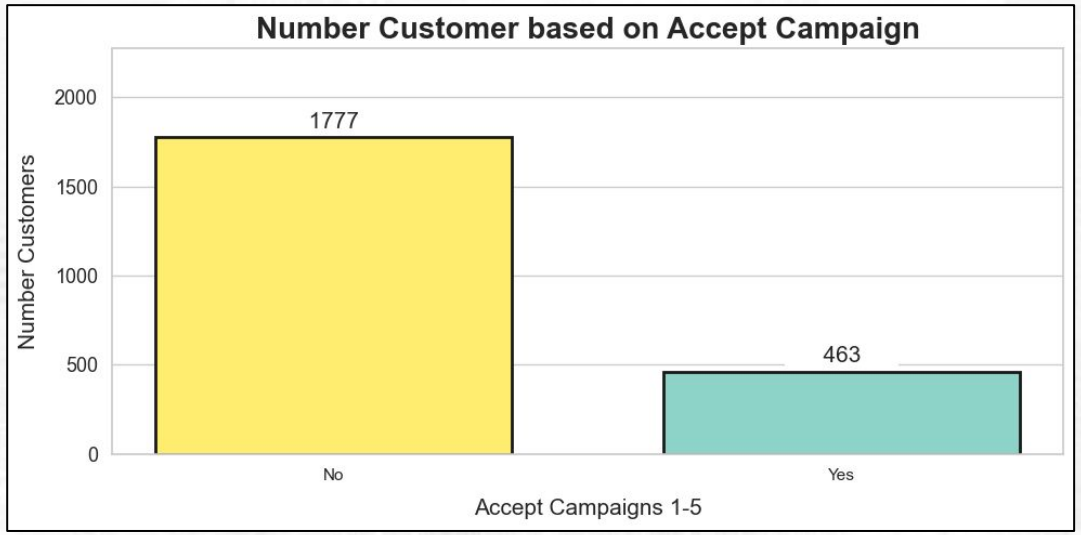
## Acceptance Rate for each Campaign (1-5)

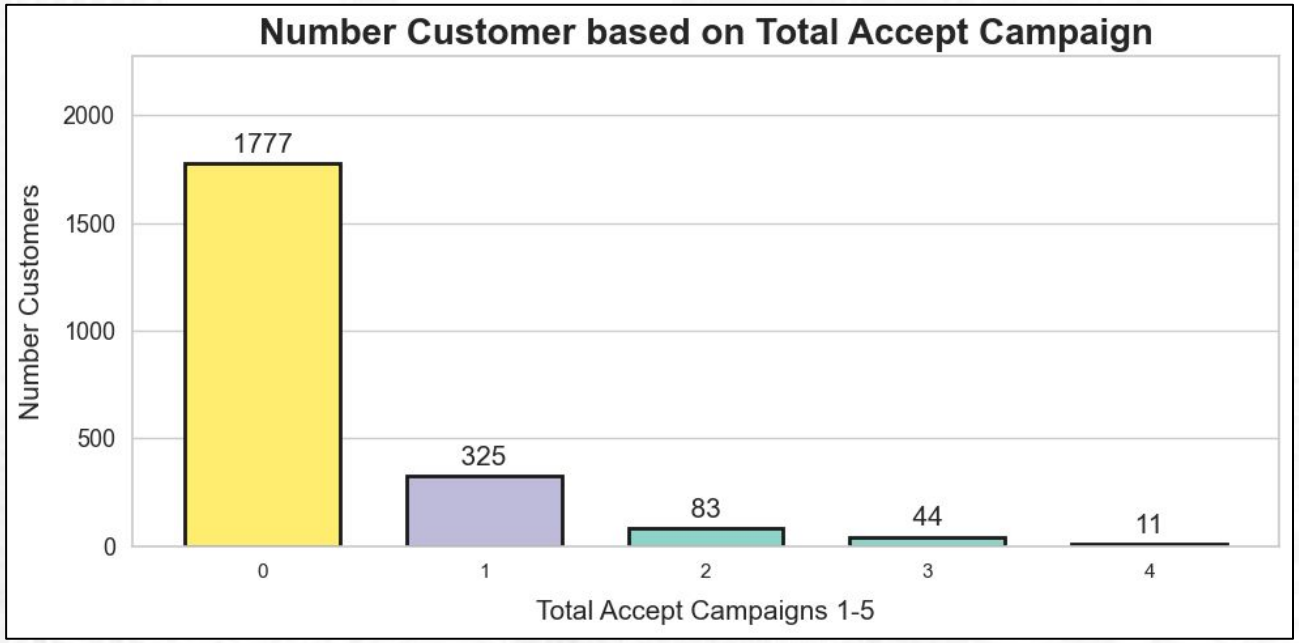



Dari **Campaign Rate** yang telah kita lakukan, terdapat **perubahan** yang terjadi dari **Campaign 1 ke Campaign 2 (menurun drastis) dan 3 (naik signifikan).** Adapun dari **Campaign 3-5** Semuanya memiliki rate yang kurang lebih **sama disekitar ~7%.** Sehingga dari perusahaan dan tim perlu melakukan **identifikasi lanjutan untuk mengetahui apakah ada pola dari pembelian customer**

# Business Insight

## Total Accept Campaign (1-5)

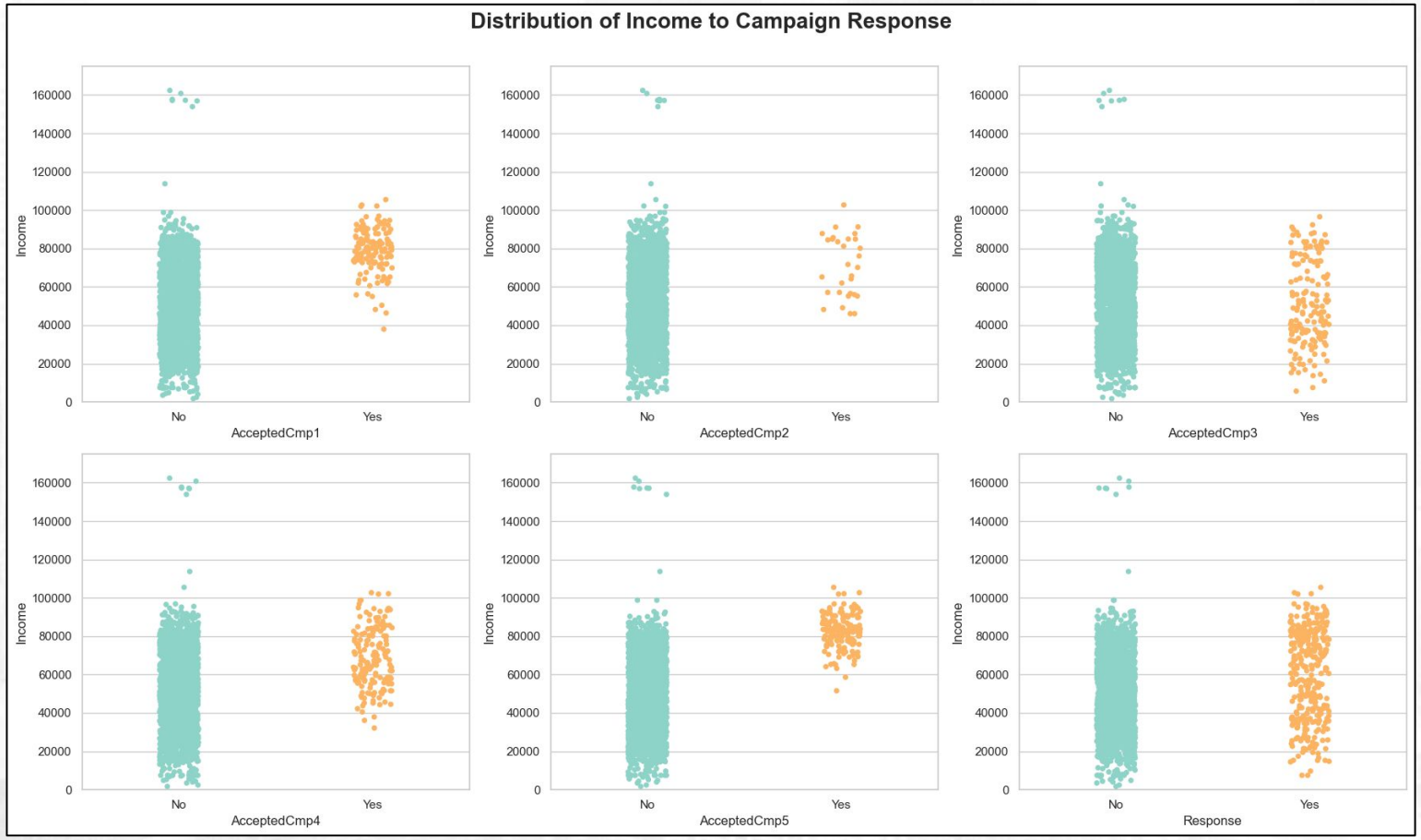



Paling banyak pada **Lima Campaign** kita adalah **0 (tidak pernah merespon),** namun ada yang sedikit **berpotensi** pada, hanya **sekali (1) atau dua kali (2) merespon** masing-masing **325 dan 83 Customers**

# Business Insight

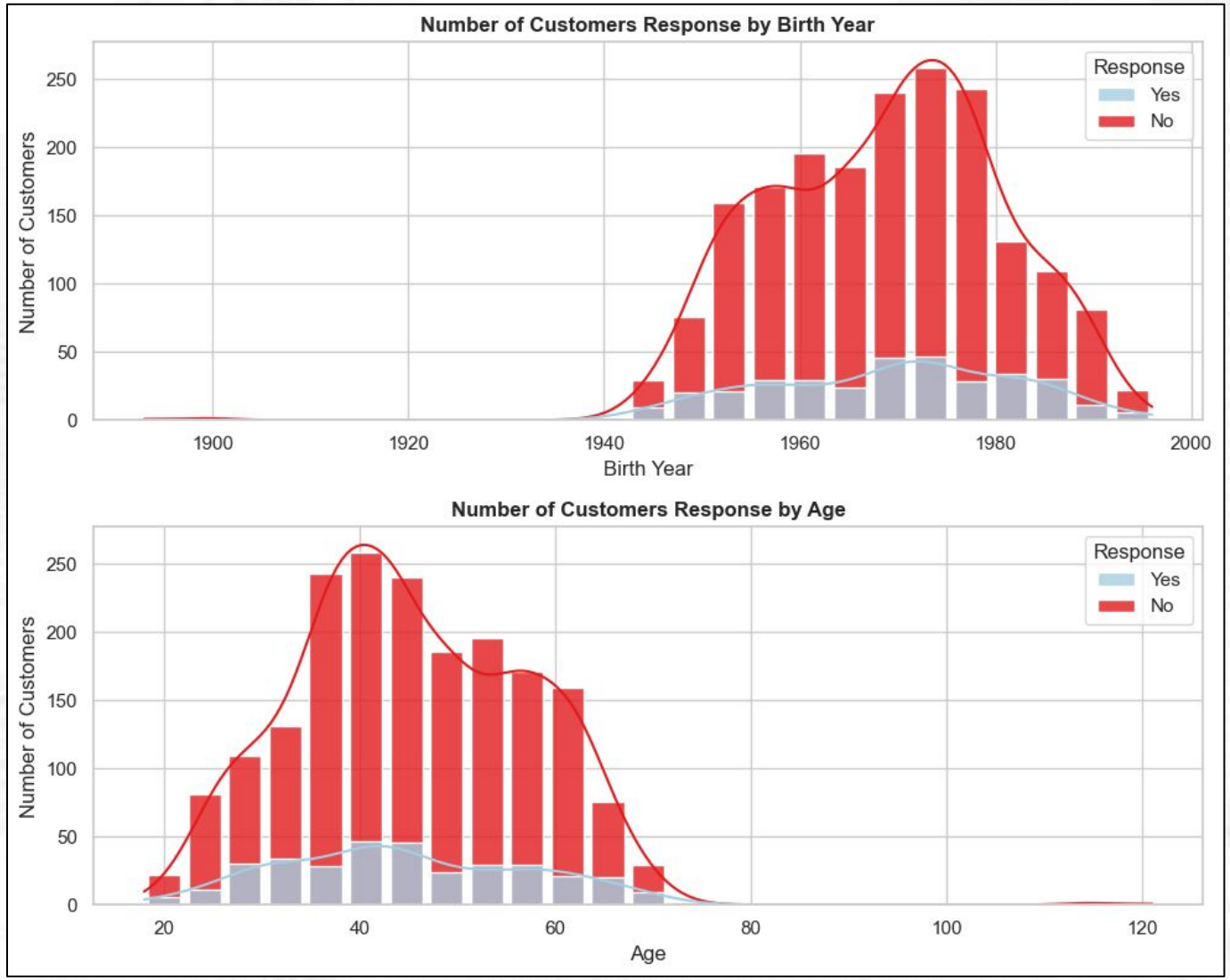
## Income vs. Campaign




**Customer** dengan **income diatas > $120.000** tidak ada yang menerima/respon campaign perusahaan. Jadi sebaiknya perusahaan **fokus melakukan campaign kepada customer dengan income dibawah < $120000.**

# Business Insight

## Birth Year / Age vs. Response



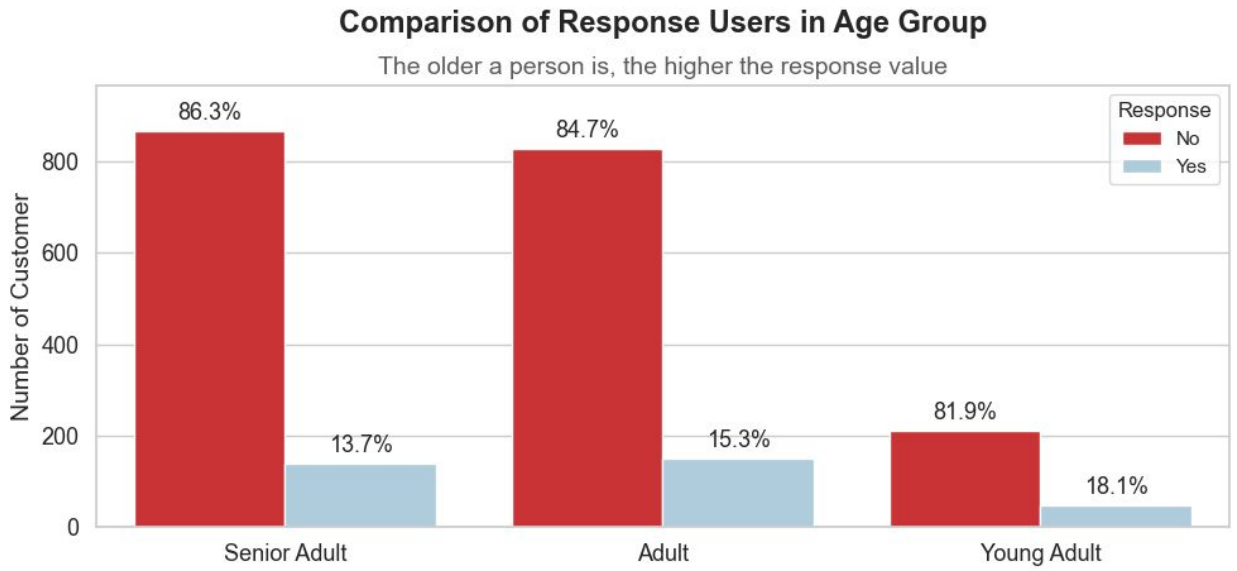

**Kategori customer** yang **menerima Response** (marketing campaign terakhir) terbanyak berasal dari **tahun lahir 1970-1975 (39-44 years old),** dan **1980-1990 (24-34 years old).**

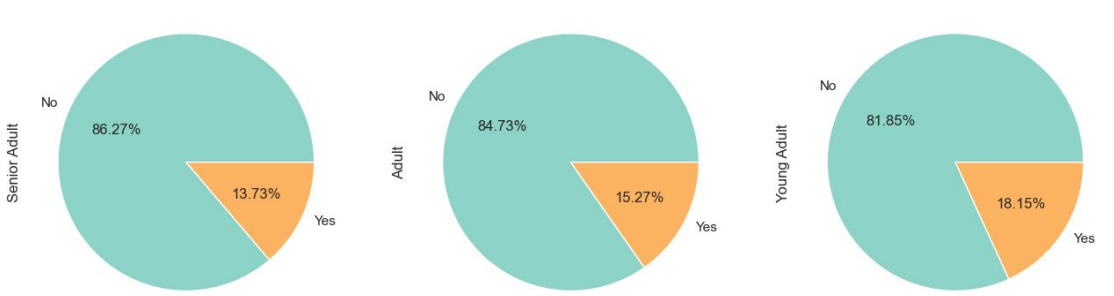
Jika perusahaan harus memprioritaskan beberapa customer saja, maka perusahaan dapat memilih customer yang lahir pada tahun tersebut untuk menawarkan sebuah campaign.

Namun tetap memperhatikan juga **No Response (Tidak Menerima)** karena pada area **tahun lahir 1970-1975 juga sangat tinggi.**

# Business Insight

## Age Group vs. Response



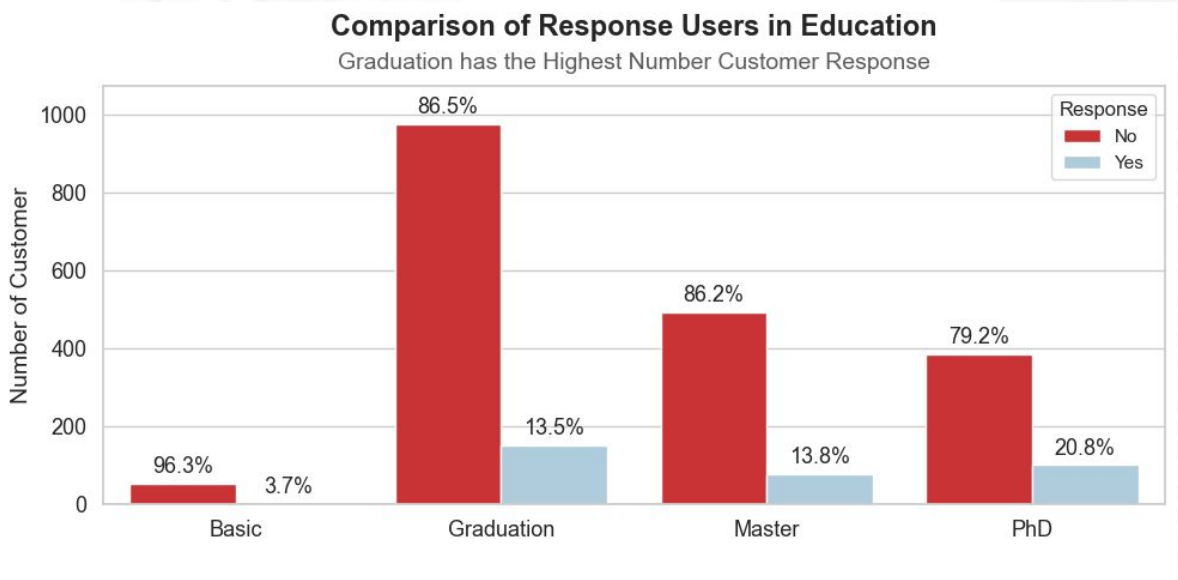



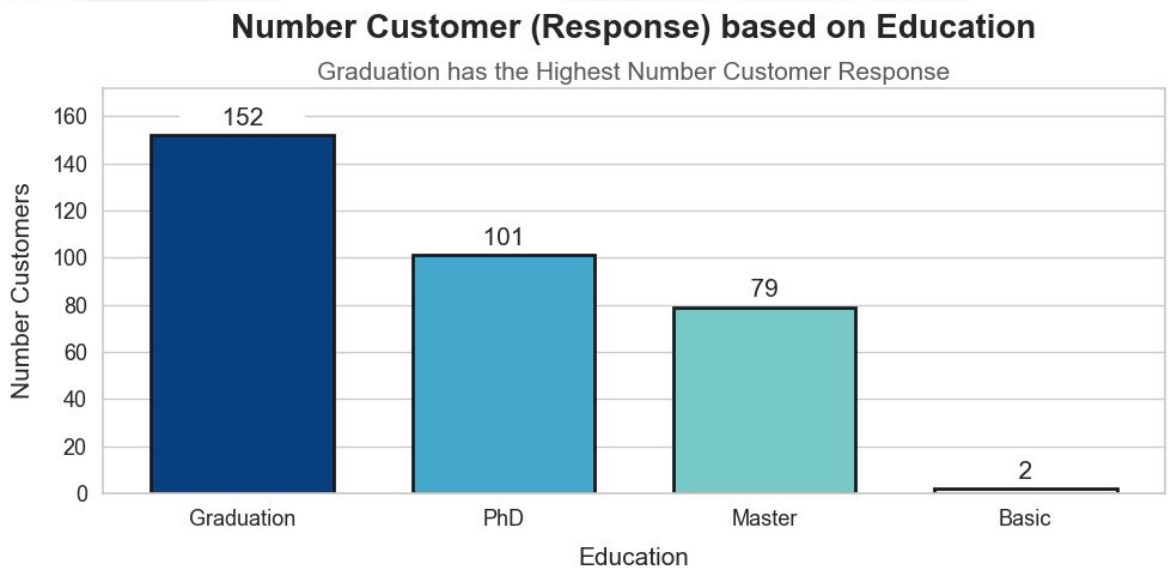

Berdasarkan chart diatas dapat dilihat bahwa kelompok umur **yang paling banyak** merespon campaign adalah **Adult dan Senior Adult** dan **yang paling rendah** adalah **Young Adult.**

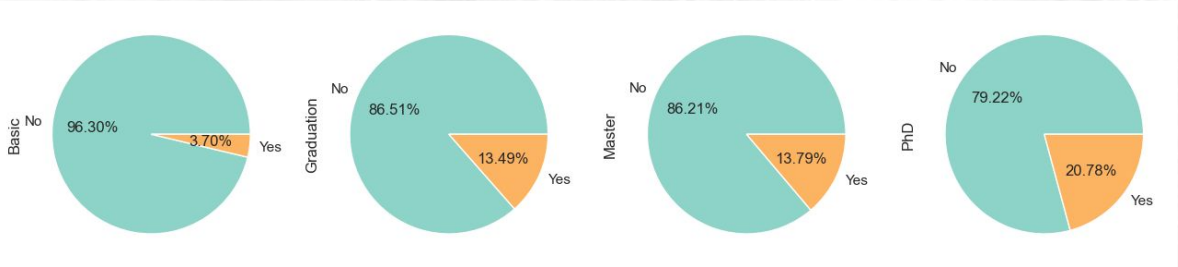
**Artinya semakin Tua seseorang maka jumlah response juga meningkat**

# Business Insight

## Education vs. Response



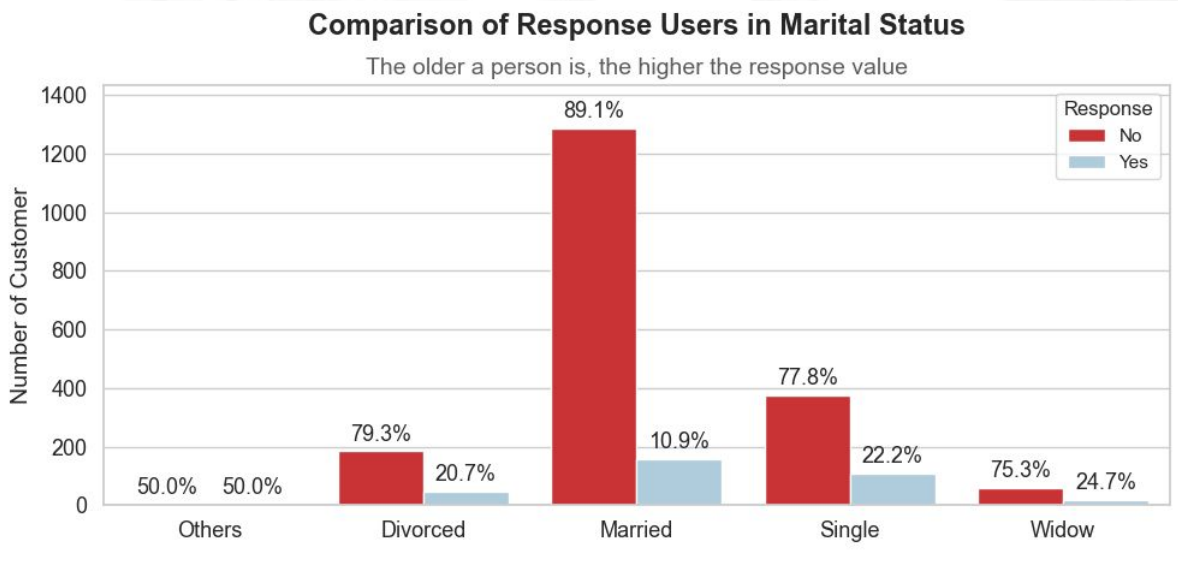



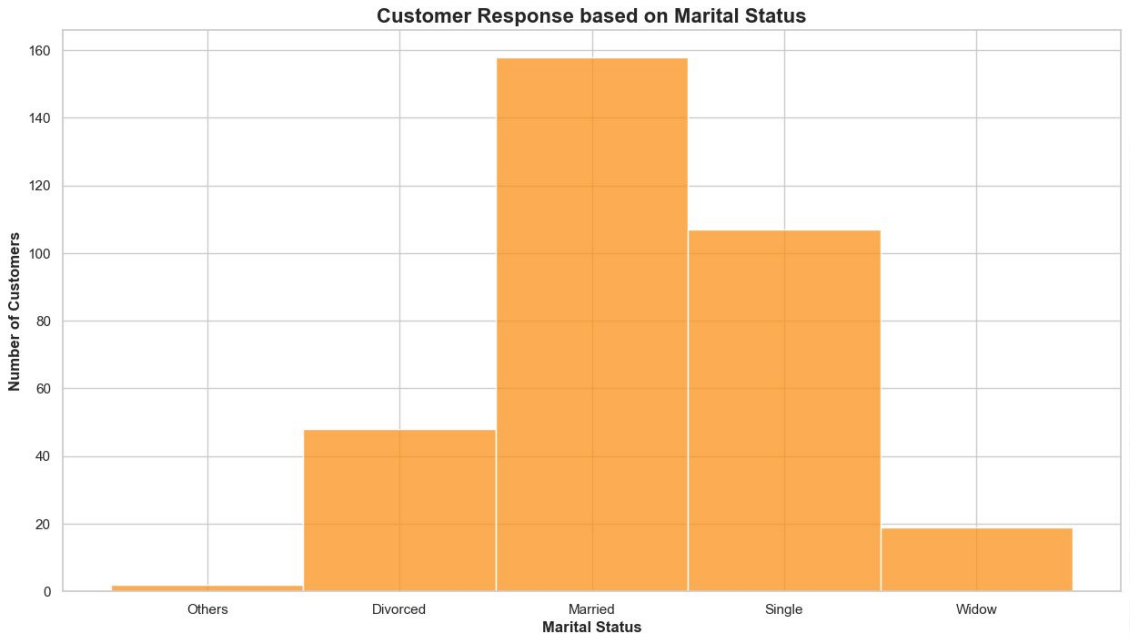



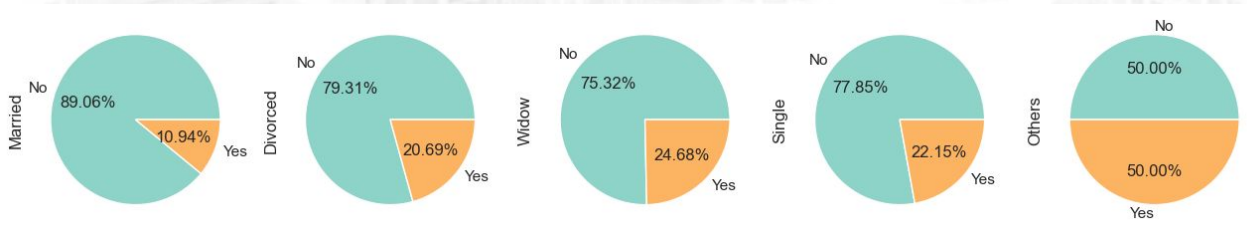

Dari visualisasi **Education**, dapat dilihat bahwa customer **yang merespon terbanyak** berasal dari customer yang memiliki edukasi **Graduation dan PhD**, sehingga marketing team dapat memfokuskan campaign ke customer yang beredukasi Graduation.

# Business Insight

## Marital Status vs. Response



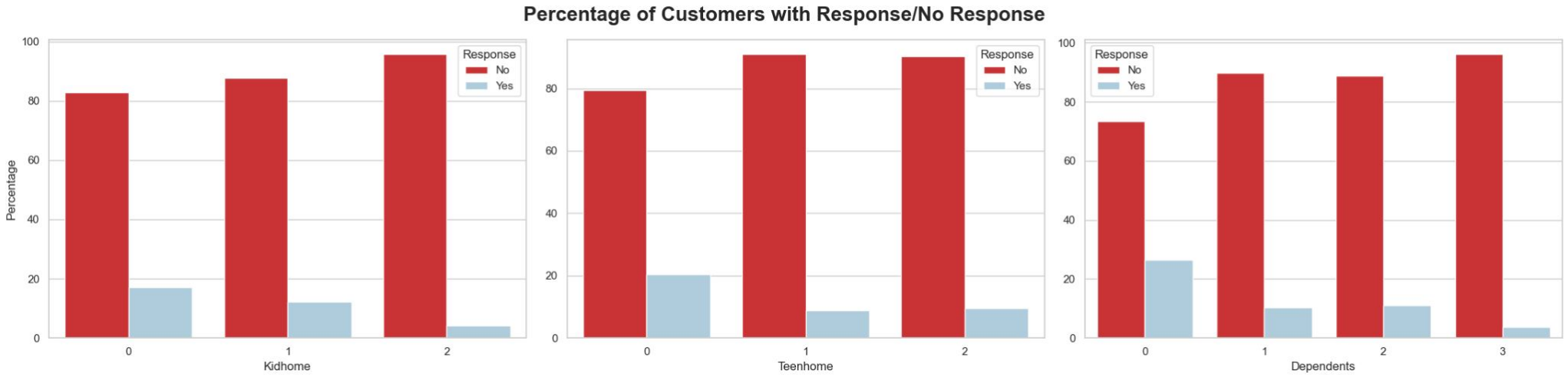



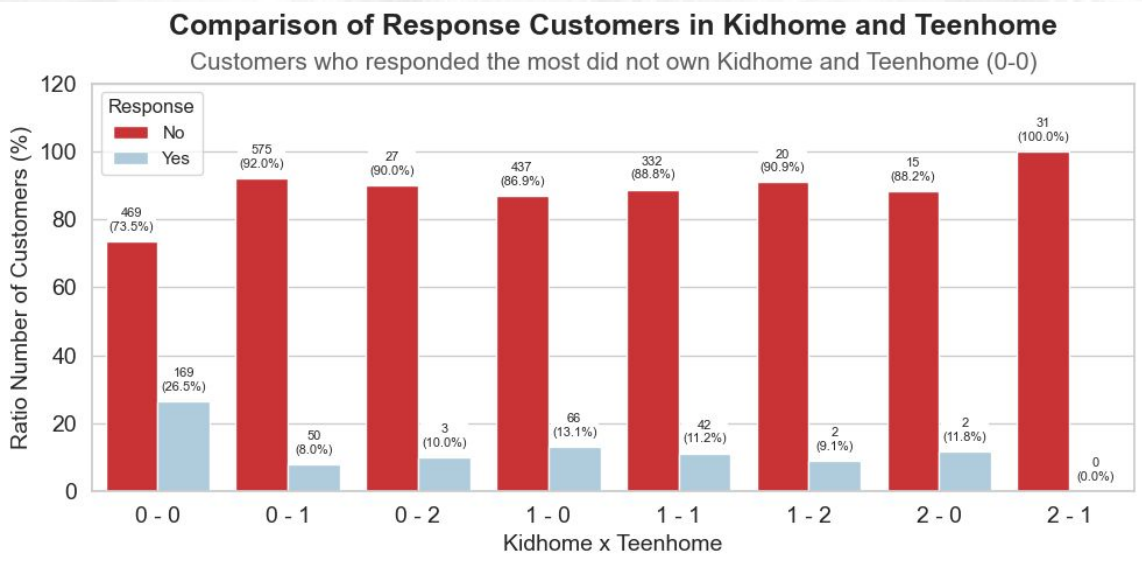




Dari visualisasi **Marital Status**, dapat dilihat bahwa customer **yang merespon terbanyak** berasal dari customer dengan status pernikahan yaitu **Married dan Single**, sehingga marketing team dapat memfokuskan campaign ke customer yang telah menikah (Married).
**Sedangkan** untuk **"Absurd"** dan **"Yolo"** pada data **"Others" sangat sedikit**

# Business Insight

## Kids and Teens (Dependents) vs. Response



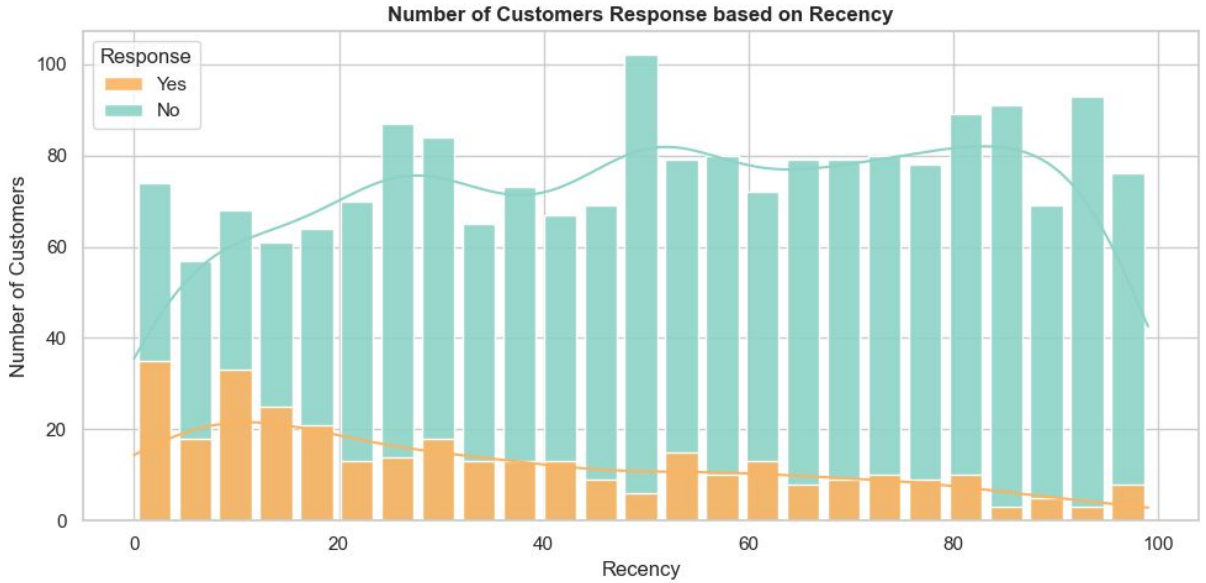



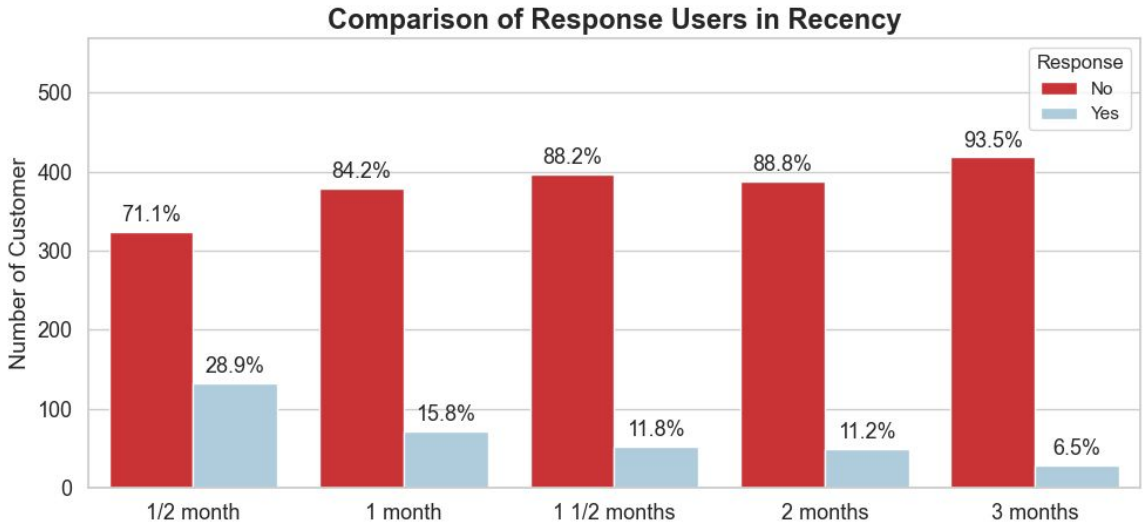

*Semakin tinggi jumlah anak/remaja yang dimiliki customer, maka semakin kecil kemungkinan customer menerima Response*

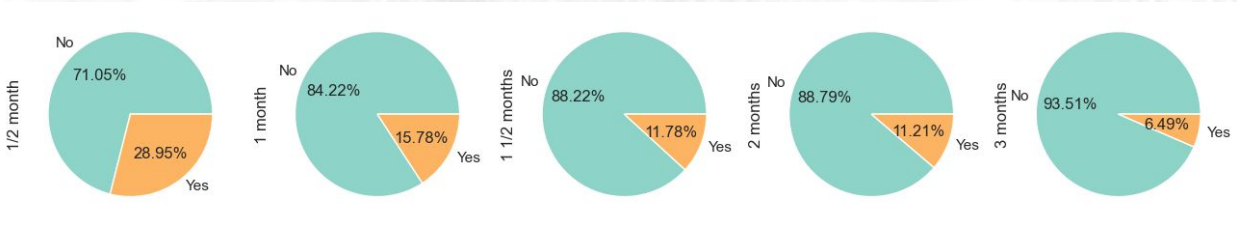
Dari visualisasi **Kids and Teens (Dependents)**, dapat dilihat bahwa customer **yang merespon terbanyak** berasal dari customer yang **tidak memiliki anak dan remaja**, sehingga marketing team dapat memfokuskan campaign ke customer yang tidak memiliki anak atau remaja.

# Business Insight

## Recency vs. Response



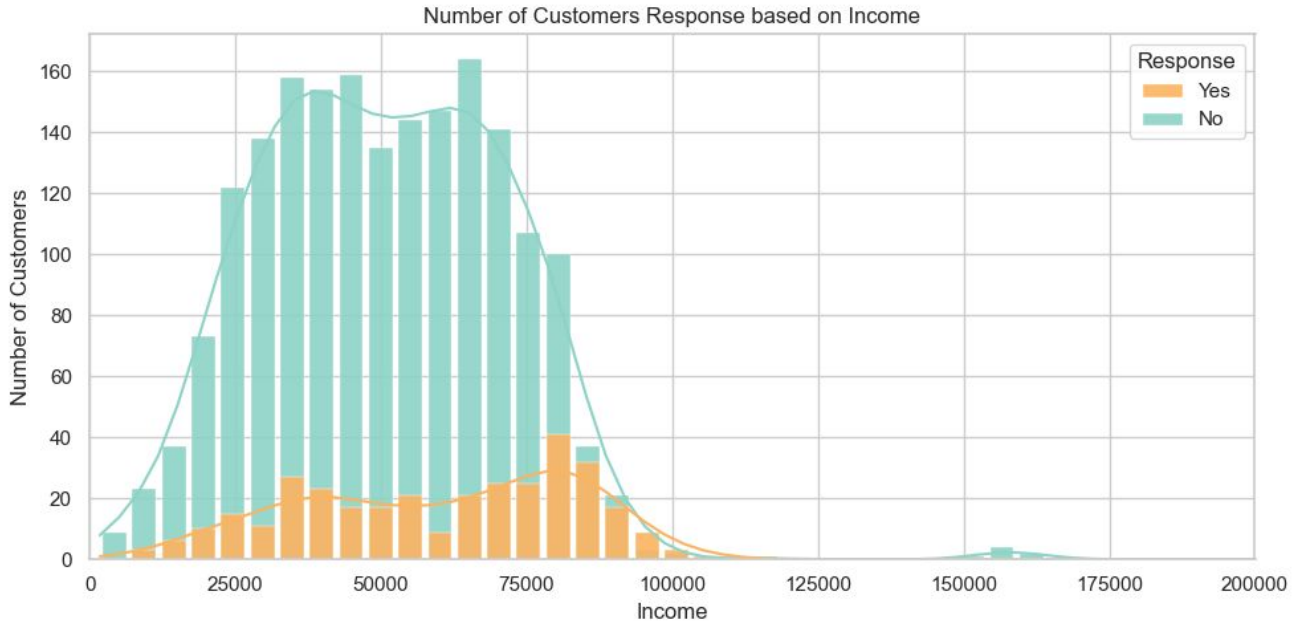



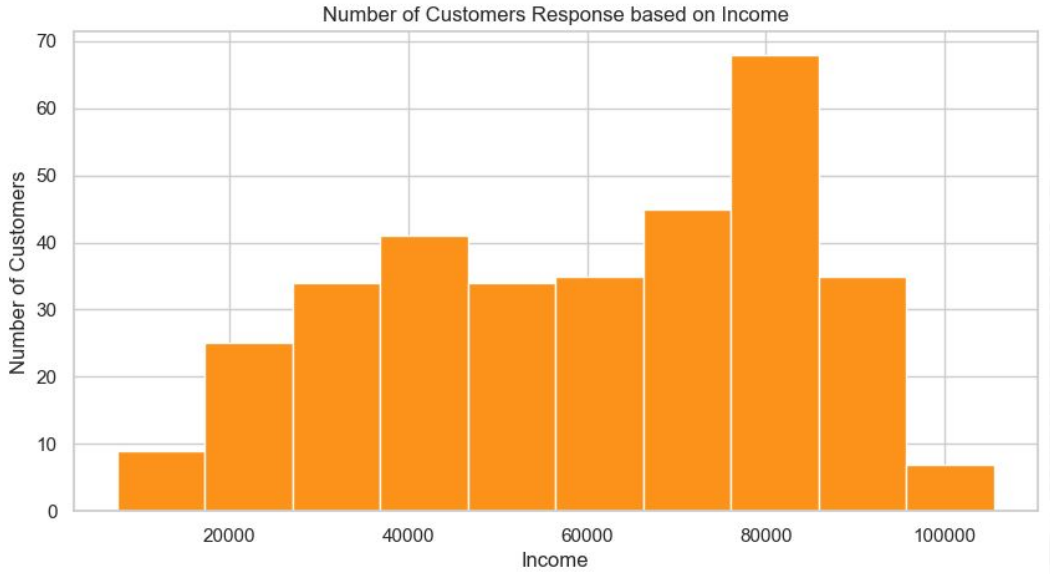

Dari visualisasi **Recency**, dapat dilihat bahwa customer **yang merespon terbanyak** berasal dari customer dengan **Recency yang rendah**, sehingga marketing team dapat memfokuskan campaign ke customer yang memiliki Recency rendah.

* ***Recency rendah*** *: waktu purchase terakhir pelanggan dengan produk.belum terlalu lama*

# Business Insight

## Income vs. Response

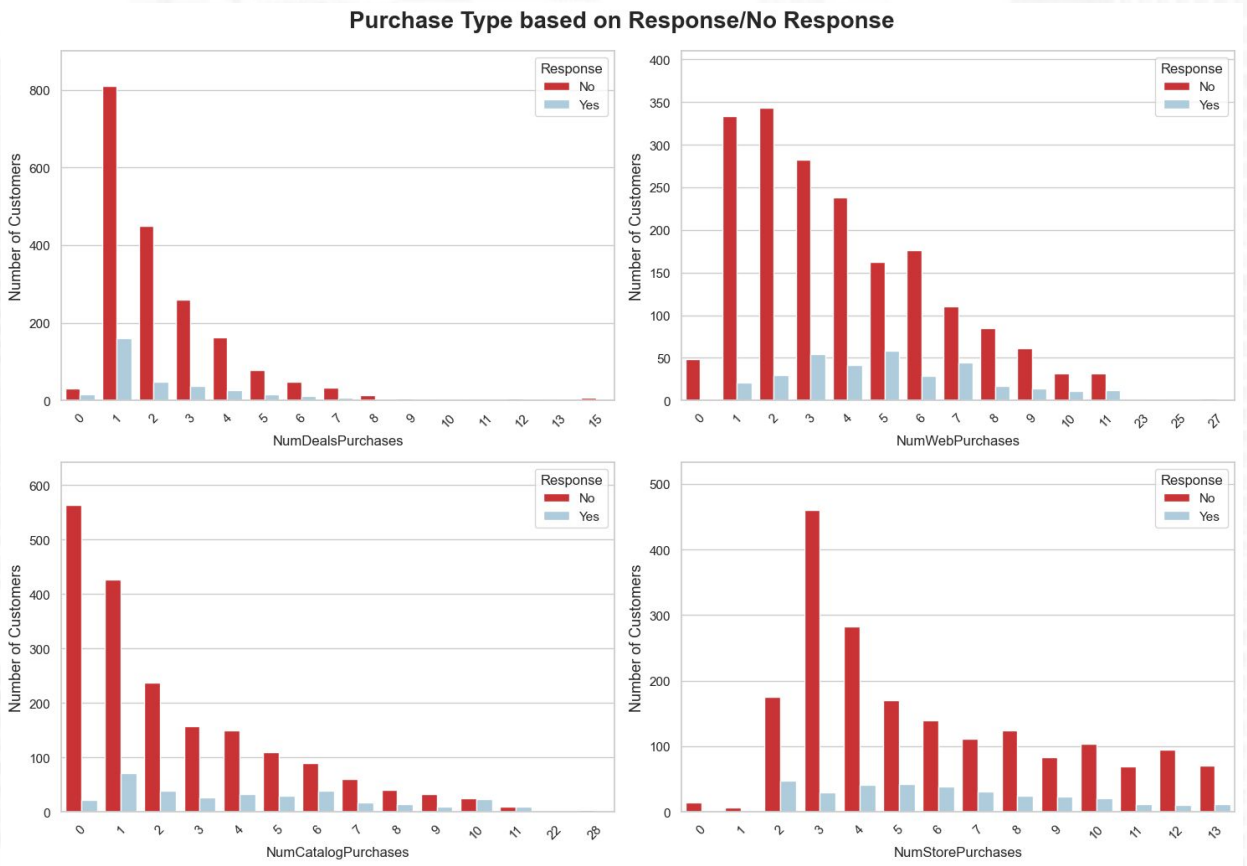


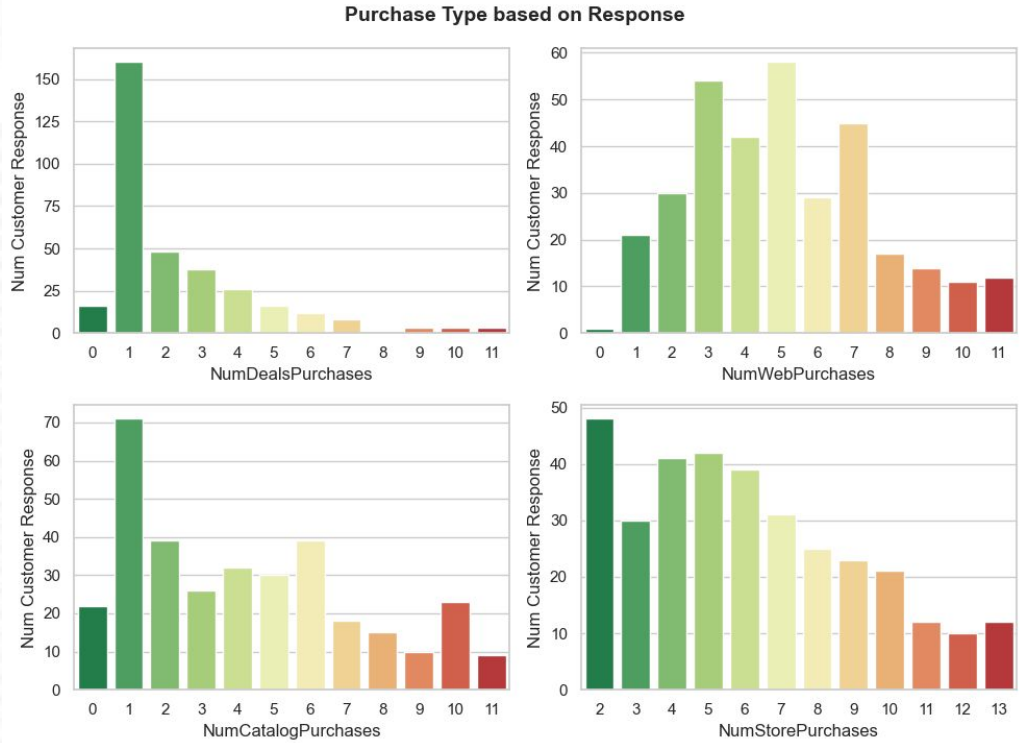




Dari visualisasi **Income**, dapat dilihat bahwa customer **yang merespon terbanyak** berasal dari customer dengan pendapatan **> $75000**, sehingga marketing team dapat memfokuskan campaign ke customer yang memiliki pendapatan diatas 75000.

# Business Insight

## Purchase Type vs. Response

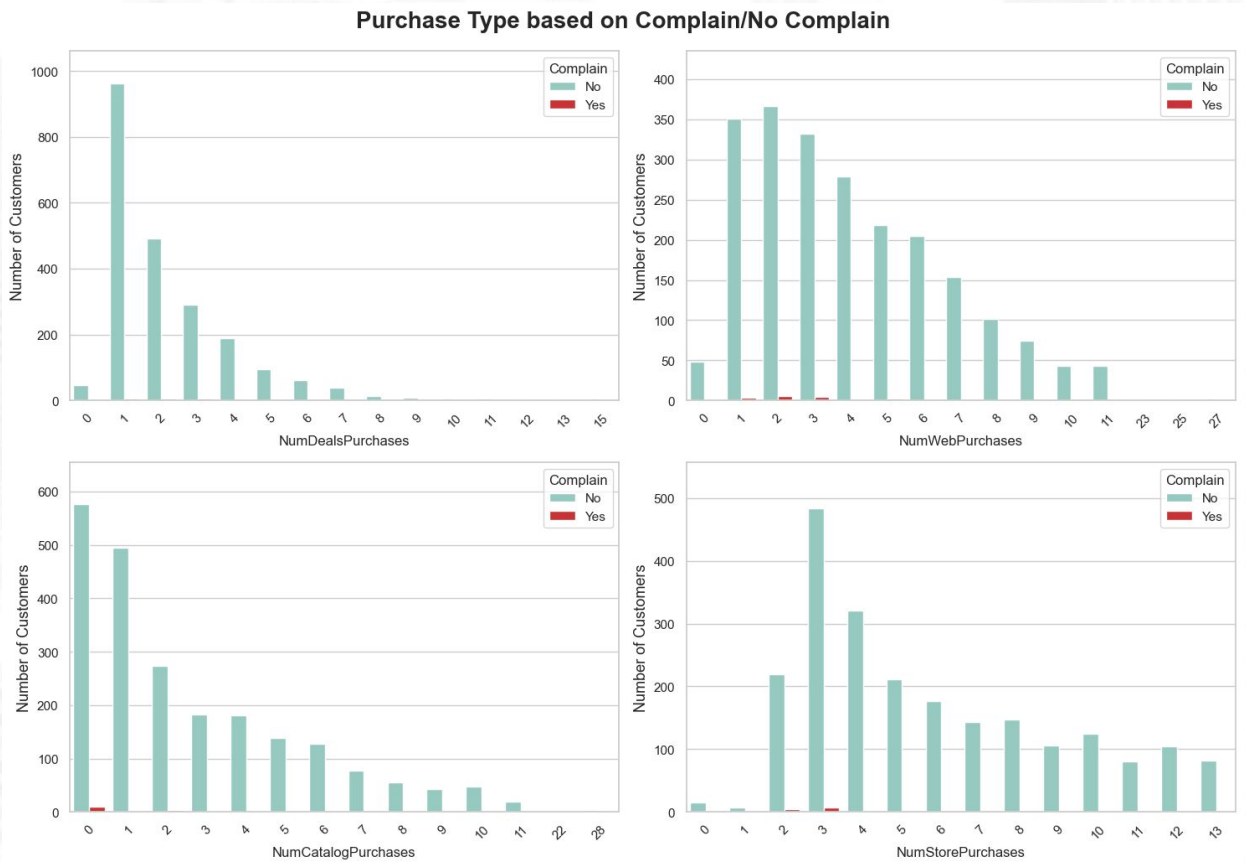


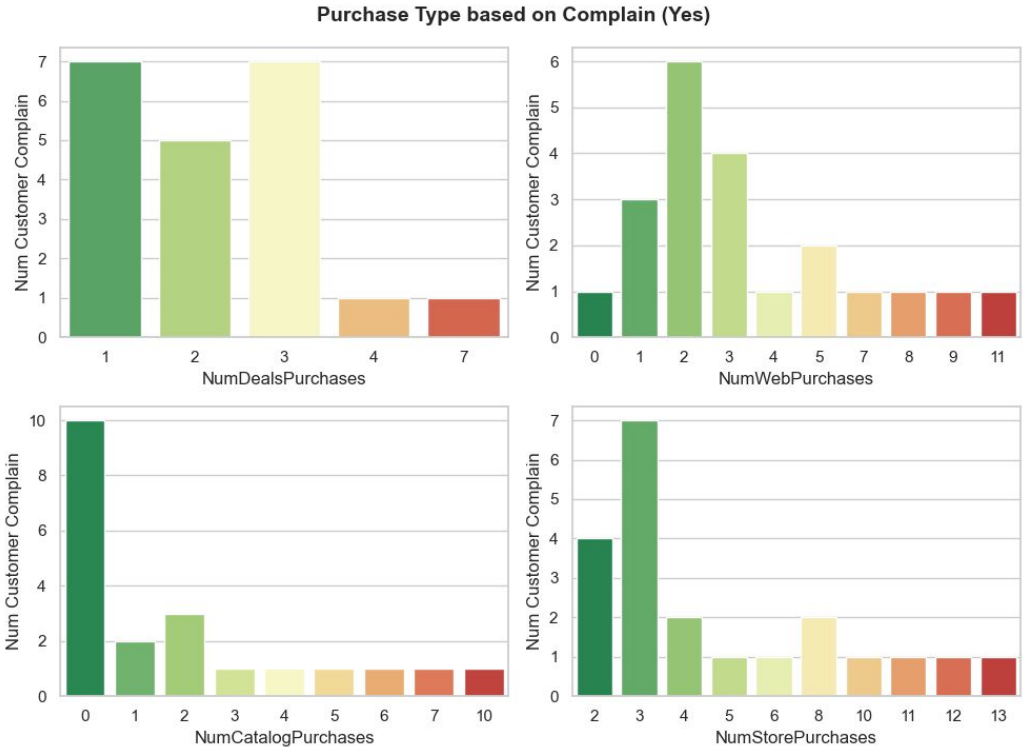




Dari visualisasi **Purchase Type**, dapat dilihat bahwa semakin sedikit pembelian yang dilakukan (baik yang menggunakan diskon ataupun yang melalui web, catalog, store), maka semakin besar kemungkinan customer untuk menerima Response, sehingga marketing team dapat memfokuskan campaign ke customer dengan jumlah pembelian yang masih sedikit.

# Business Insight

## Purchase Type vs. Complain

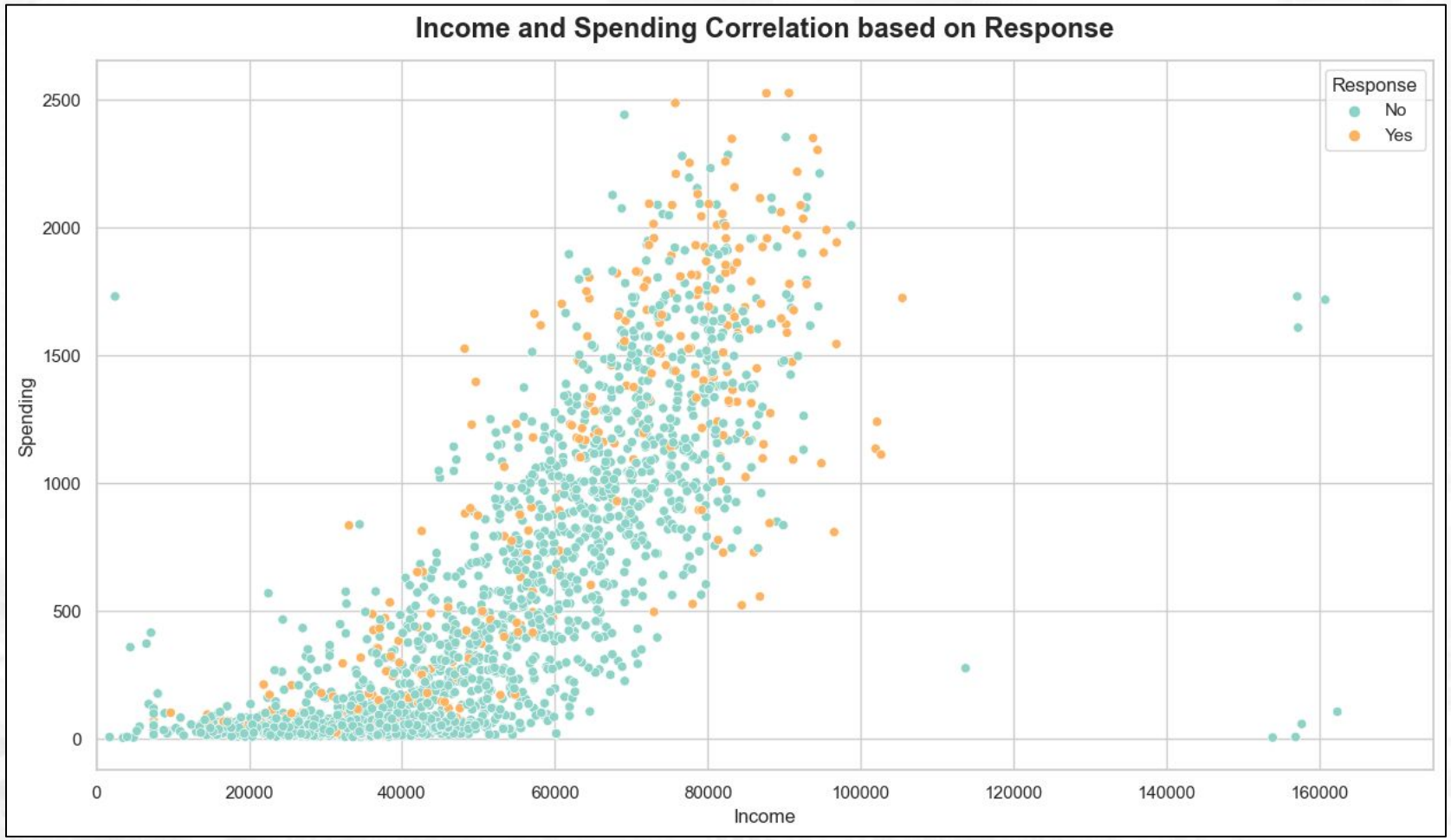



Dari visualisasi **Purchase Type**, dapat dilihat bahwa customer **yang complain terbanyak** berasal dari customer dengan Purchase rendah, sehingga perusahaan sebaiknya meningkatkan kualitas pelayanan untuk keseluruhan customer dengan Purchase yang rendah maupun tinggi.

# Business Insight

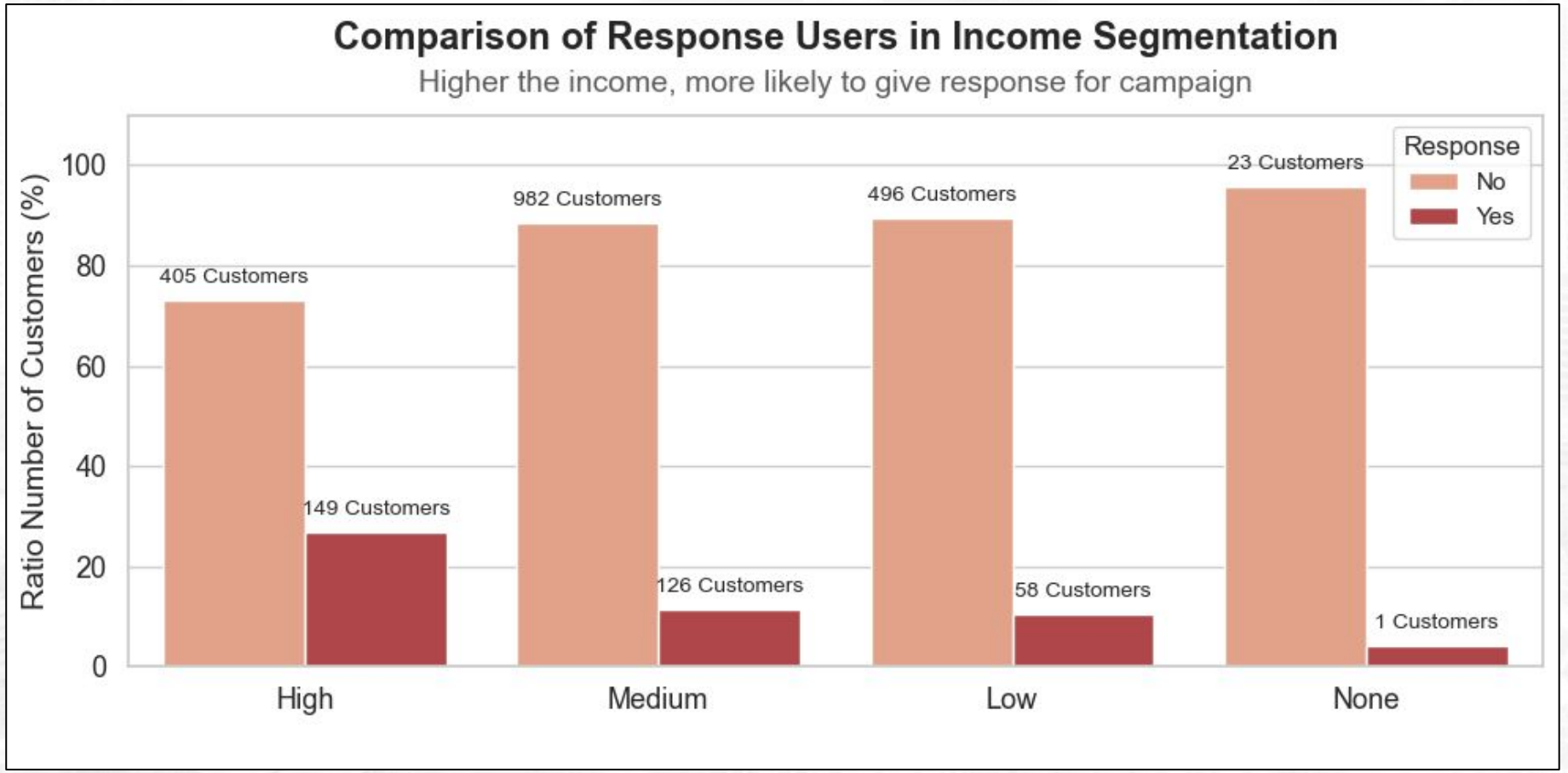
## Income x Spending for Response




**Income dan Spending** memiliki korelasi positif pada response, dimana **semakin tinggi nilai income dan spending semakin besar tingkat respon** sehingga fitur income dan spending perlu dipertahankan

# Business Insight

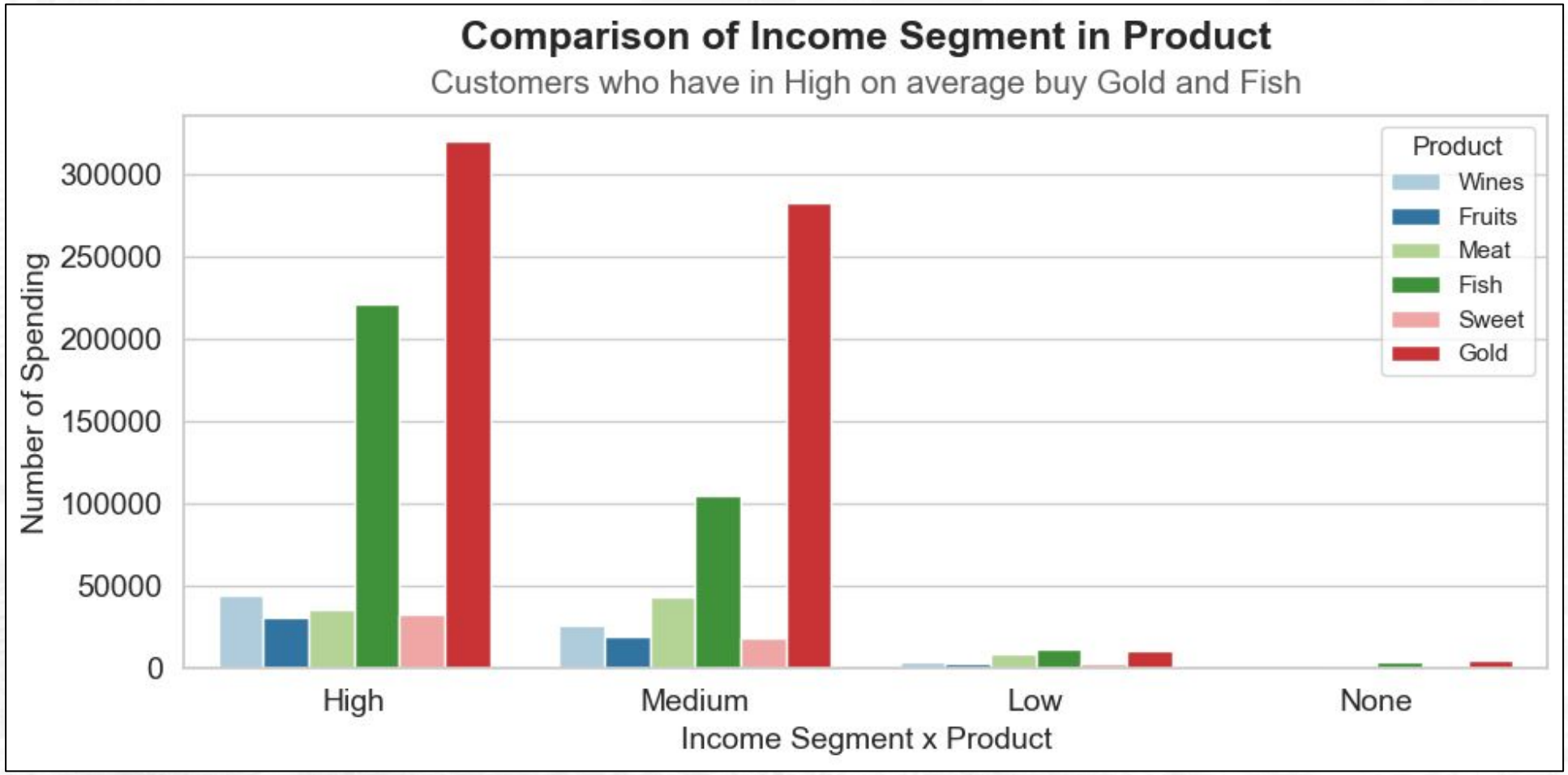
## Income Segmentation x Response




**Customer** yang merespon campaign cenderung memiliki **income yang lebih tinggi,** terbukti dari customer yang memiliki **income "High" level  merespon campaign lebih banyak.**

# Business Insight

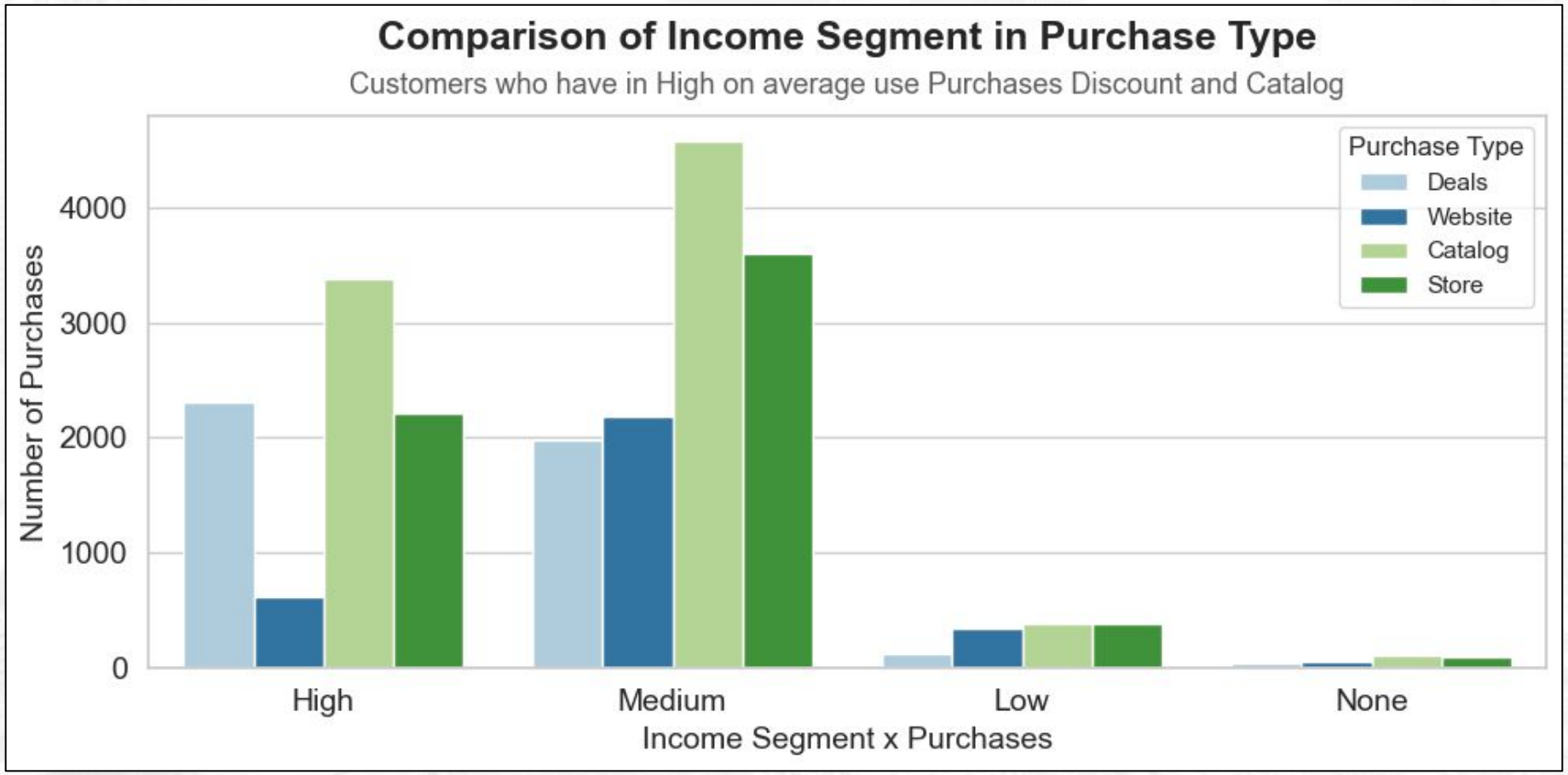
## Income Segmentation x Product Type




Customer dengan **income High dan Medium** lebih suka dengan produk **Gold dan Fish**. Oleh karena itu, jika ingin membuat campaign disarankan untuk memberikan campaign produk Gold dan Fish untuk customer dengan income tersebut.

# Business Insight

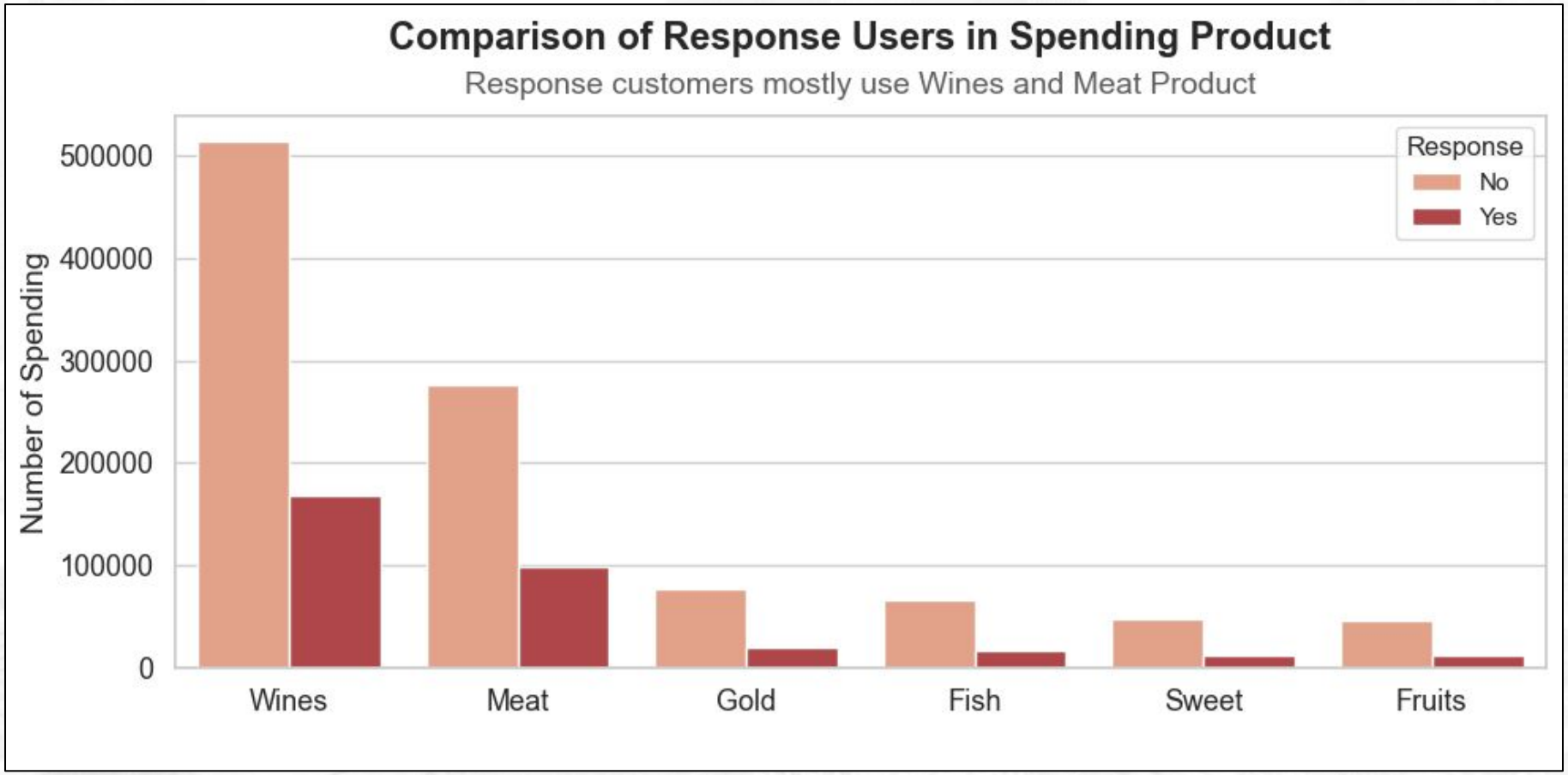
## Income Segmentation x Purchase Type




Untuk pembuatan campaign, disarankan berbentuk **Catalog atau Diskon** dan lebih banyak diarahkan ke customer dengan income **High / Medium**. Untuk opsi kedua, bisa langsung dibuat banner/booth pada Store.

# Business Insight

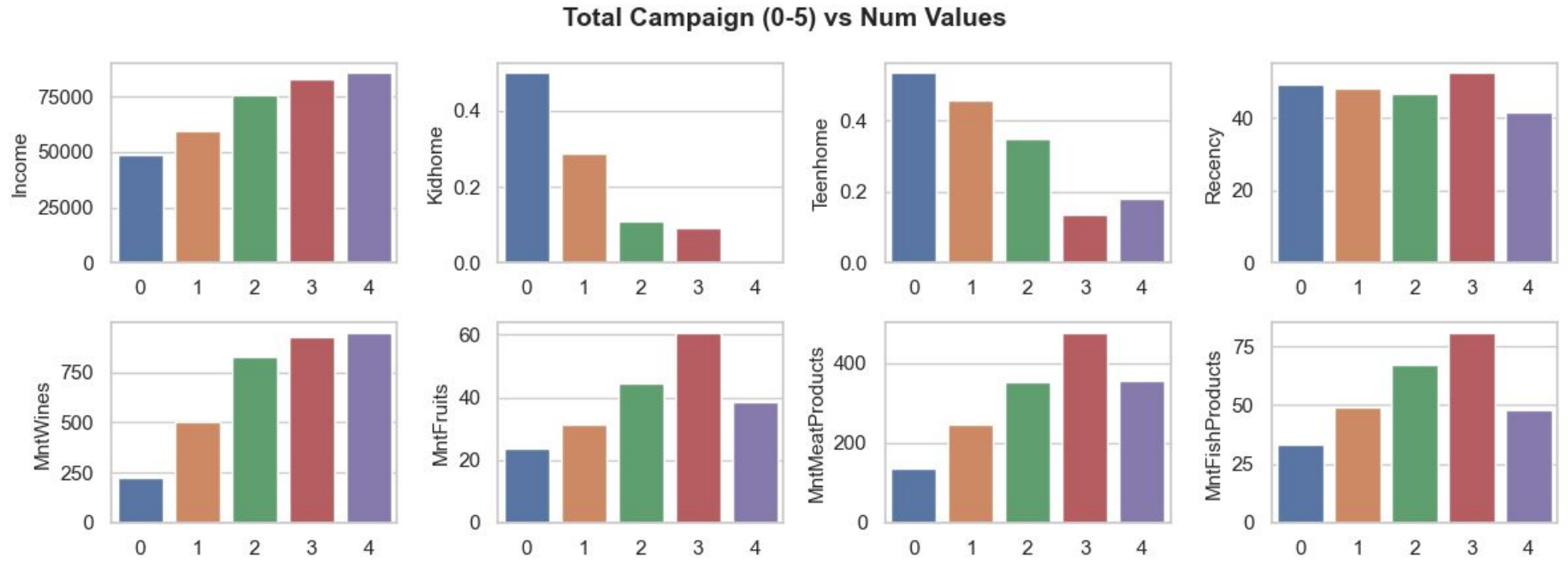
## Product Type x Response




**Customer yang merespon campaign** cenderung lebih banyak membeli **Wines dan Meat products**. Sehingga untuk campaign selanjutnya produk Wines dan Meat menjadi rekomendasi produk utama untuk customer yang merespon.

# Business Insight

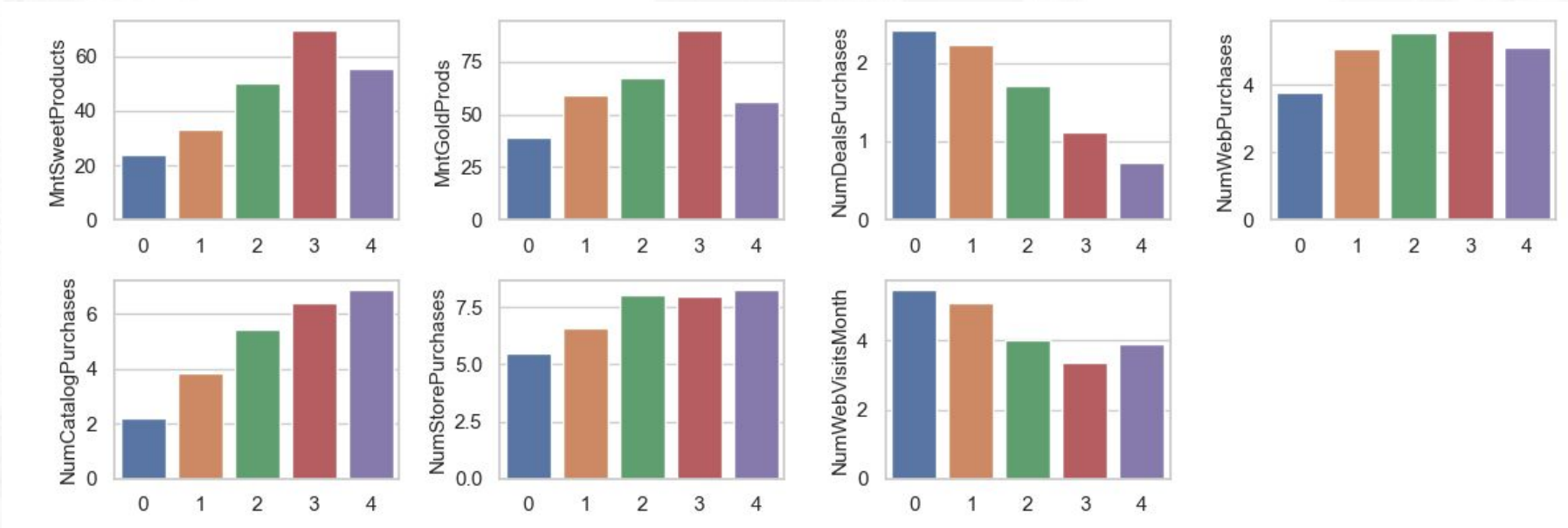
## Total Campaign vs Variable




- **Income:**
  Customer yang menerima total 4 campaign cenderung memiliki income yang lebih tinggi.
- **Kidhome:**
  Customer yang tidak menerima campaign sama sekali memiliki kidhome lebih banyak.
- **Teenhome:**
  Customer yang tidak menerima campaign sama sekali memiliki Teenhome lebih banyak.
- **Recency:**
  Customer yang menerima total 3 campaign cenderung memiliki recency (total hari terakhir berbelanja) yang lebih lama.
- **MntWines:**
  Customer yang menerima total 4 campaign cenderung membeli produk Wines lebih banyak.
- **MntFruits:**
  Customer yang menerima total 3 campaign cenderung membeli produk Fruits lebih banyak.
- **MntMeatProducts:**
  Customer yang menerima total 3 campaign cenderung membeli produk Meat lebih banyak.
- **MntFishProducts:**
  Customer yang menerima total 3 campaign cenderung membeli produk Fish lebih banyak.

# Business Insight

## Total Campaign vs Variable




Berikut adalah insights yang diperoleh berdasarkan *Total Campaign vs Variable:*

- **MntSweetProducts:**
  Customer yang menerima total 3 campaign cenderung membeli produk Sweet lebih banyak.
- **MntGoldProducts:**
  Customer yang menerima total 3 campaign cenderung membeli produk Gold lebih banyak.
- **NumDealsPurchases:**
  Customer yang tidak menerima campaign sama sekali lebih sering membeli produk menggunakan diskon.
- **NumWebPurchases:**
  Customer yang menerima total 2 dan 3 campaign cenderung lebih sering membeli produk melalui web.
- **NumCatalogPurchases:**
  Customer yang menerima total 4 campaign cenderung lebih sering membeli produk melalui katalog.
- **NumStorePurchases:**
  Customer yang menerima total 4 campaign cenderung lebih sering membeli produk melalui toko.
- **NumWebVisitsMonths:**
  Customer yang tidak menerima campaign sama sekali lebih sering melakukan web visit pada bulan terakhir.

Rakamin Academy

# Stage 2

*Data Pre-processing*

# DATA PRE-PROCESSING

## Classification

**Handling Duplicated Rows**

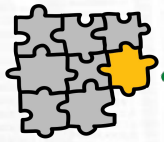
0 Duplicated Row

**Handling Invalid Values**

Dt_Customer,
Marital_Status,
Education

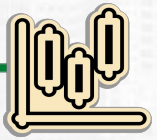
**Data Splitting**

Data Train : 75% (1680 rows)
Data Test : 25% (560 rows)



**Handling Missing Values**

Income : 24 rows
MICE with LightGBM



**Handling Outliers**

Manually Trimmed

**Feature Extraction**

15 feature

**Feature Transformation**

Yeo-Johnson

**Feature Encoding**

Label Encoding,
One Hot Encoding

**Feature Selection**

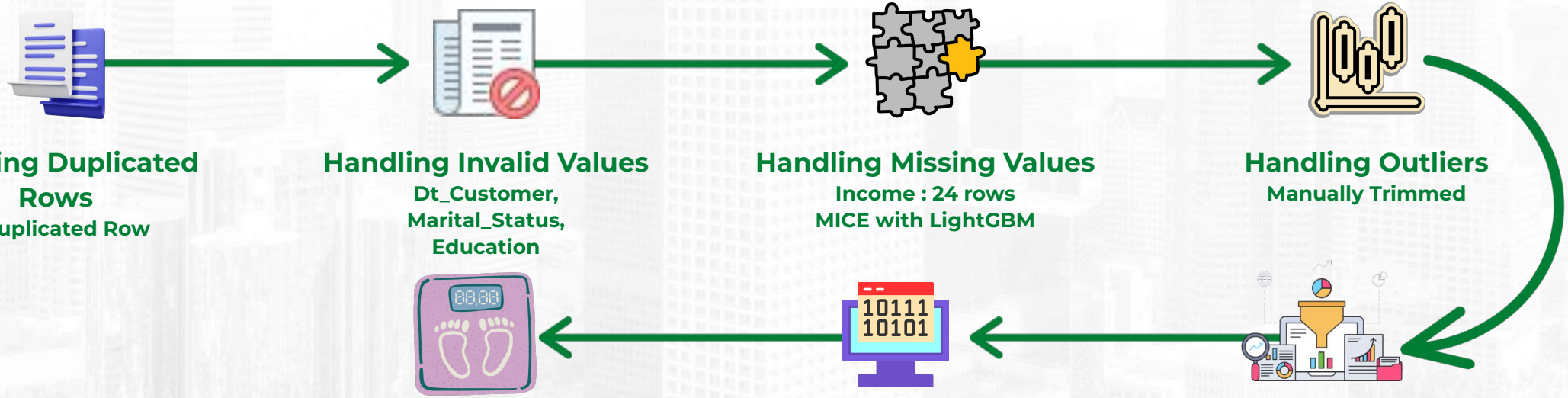
Anova,
Variance, MI,
SelectkBest,
Multicollinearity

**Handling Imbalance**

SMOTE

# DATA PRE-PROCESSING

## Clustering



**Handling Duplicated Rows**
0 Duplicated Row

**Handling Invalid Values**
Dt_Customer,
Marital_Status,
Education

**Handling Missing Values**
Income : 24 rows
MICE with LightGBM

**Handling Outliers**
Manually Trimmed

**Feature Extraction**
15 feature

**Feature Encoding**
Label Encoding,
One Hot Encoding

**Feature Scalling**
Standar Scaller

# Handling Duplicate Rows

```
df.duplicated().sum()
```

```
0
```

```
print(f"Data Frame Dimension Before Duplicate Removal: {df.shape}")
df = df.drop_duplicates().reset_index(drop=True)
print(f"Data Frame Dimension After Duplicate Removal: {df.shape}")
```

```
Data Frame Dimension Before Duplicate Removal: (2240, 29)
Data Frame Dimension After Duplicate Removal: (2240, 29)
```

```
df.duplicated(subset=["ID"]).sum()
```

```
0
```

**Kesimpulan**

- Berdasarkan hasil pengecekan pada dataset tidak ditemukan baris data yang memiliki duplikat, sehingga kita tidak perlu melakukan handling duplicated data
- Pada pengecekan duplikat subset untuk ID tidak ditemukan ada nya ID customer yang sama

# Handling Invalid Values

Beberapa hal yang dilakukan pada proses Handling Invalid Values diantaranya:

- **Melakukan konversi data Date**

```
df["Dt_Customer"] = pd.to_datetime(df["Dt_Customer"])
```

Untuk mempermudah dalam **proses feature extraction/engineering** maka untuk data yang mengandung datetime akan dilakukan **konversi ke format datetime pandas**

- **Melakukan penyederhanaan Marital_Status**

```
# Mengganti kategori 'Widow', 'Alone', 'Absurd', 'YOLO'
df['Marital_Status'] = df['Marital_Status'].replace(['Widow', 'Alone', 'Absurd', 'YOLO'],'Single')
# Mengganti kategori 'Together' menjadi 'Married'
df['Marital_Status'] = df['Marital_Status'].replace(['Together'],'Married')
```

```
df['Marital_Status'].unique()
```

```
array(['Single', 'Married', 'Divorced'], dtype=object)
```

Perlu dilakukan replace data / menyatukan yang memiliki arti yang sama agar mengurangi jumlah dimensi maupun redudansi pada data yang dirincikan sebagai berikut:

- Mengganti kategori **Widow**, **Alone**, **Absurd**, **YOLO** menjadi **Single**
- Mengganti kategori **Together** menjadi **Married**
- Mempertahankan kategori **Divorced**

# Handling Invalid Values

Beberapa hal yang dilakukan pada proses Handling Invalid Values diantaranya:

- **Melakukan penyederhanaan Education**

```
# Levels : Basic - Graduation - 2n Cycle - Master - PhD
df['Education'] = df['Education'].replace(['2n Cycle'],'Master')
```

Pada kategori **2n Cycle** dan **Master** akan dilakukan replace data / menyatukan yang memiliki arti yang sama juga, sehingga untuk baris yang memiliki kategori **2n Cycle** akan dihapus dan digantikan dengan kategori **Master**

**Marital_status dan Education setelah dilakukan penyederhanaan**

| | ID | Year_Birth | Education | Marital_Status |
|---|---|---|---|---|
| 0 | 5524 | 1957 | Graduation | Single |
| 1 | 2174 | 1954 | Graduation | Single |
| 2 | 4141 | 1965 | Graduation | Married |
| 3 | 6182 | 1984 | Graduation | Married |
| 4 | 5324 | 1981 | PhD | Married |

# Data Splitting

Rakamin Academy

Agar menghindari **data leakage (kebocoran data)**, Data preparation harus dilakukan **fit** dengan **training dataset** saja. Artinya, setiap koefisien atau model yang disiapkan untuk proses penyiapan data hanya boleh menggunakan deretan data dalam training dataset.

**split training set and testing set** → **75:25**

```
# splitting tha data
df_train, df_test = train_test_split(df, test_size=0.25, stratify=df[['Response']], random_state=42)
df_train.reset_index(drop=True, inplace=True)
df_test.reset_index(drop=True, inplace=True)

print(df_train.shape)
print(df_test.shape)
```

```
(1680, 29)
(560, 29)
```

**output:**

(1680, 29) (560, 29)

# Handling Missing Values

Missing values status: True

| | Total Null Values | Percentage | Data Type | NULL Train | NULL Test |
|---|---|---|---|---|---|
| **Income** | 24 | 1.071 | int64 | 20 | 4 |

Pada proses handling missing values untuk kolom **Income** ada beberapa metode yang dapat dilakukan diantaranya:

- **Drop Rows Missing Values**
  Drop rows pada missing values tidak kita dilakukan karena data yang dimiliki terbatas
- **Imputation Median**
  Proses imputasi dilakukan dengan menggunakan **median**, karena distribusi data pada kolom **Income** berbentuk **Highly Positively Skewed** dengan menggunakan fungsi **FillNa** or **SimpleImputer**
- **Multivariate Approach**
  - Melakukan **transform** beberapa **kolom object/string**, karena penggunaan **multivariate approch** memerlukan semua **kolom numeric (Feature Encoding/Drop Character)**
    - **Metode yang digunakan** diantaranya :
      a. **KNNImputer** or **K-Nearest Neighbor**
      b. **MICE** or **Multiple Imputation by Chained Equation**
      Menggunakan **IterativeImputer** or **LightGBM**
- Choice Determination:

Pada proses handling missing values ini kita menggunakan Imputation using **MICE with LightGBM**

**Imputation using MICE with LightGBM**

```
import miceforest as mf

df_ma_train_amp = mf.ampute_data(df_ma_train, perc=0.25,random_state=1991)

# Train

# Create kernel.
kds = mf.ImputationKernel(
  data = df_ma_train,
  save_all_iterations=True,
  random_state=1991
)

# Run the MICE algorithm
kds.mice(iterations=5, n_estimators=50)

# Return the completed dataset.
df_imputed_train = kds.complete_data()
df_train["Income"] = df_imputed_train["Income"].copy()
```

```
# Test
new_data_imputed = kds.impute_new_data(df_ma_test)
# Return a completed dataset
df_imputed_test = new_data_imputed.complete_data(0)
df_test["Income"] = df_imputed_test["Income"].copy()
```

**Kesimpulan**

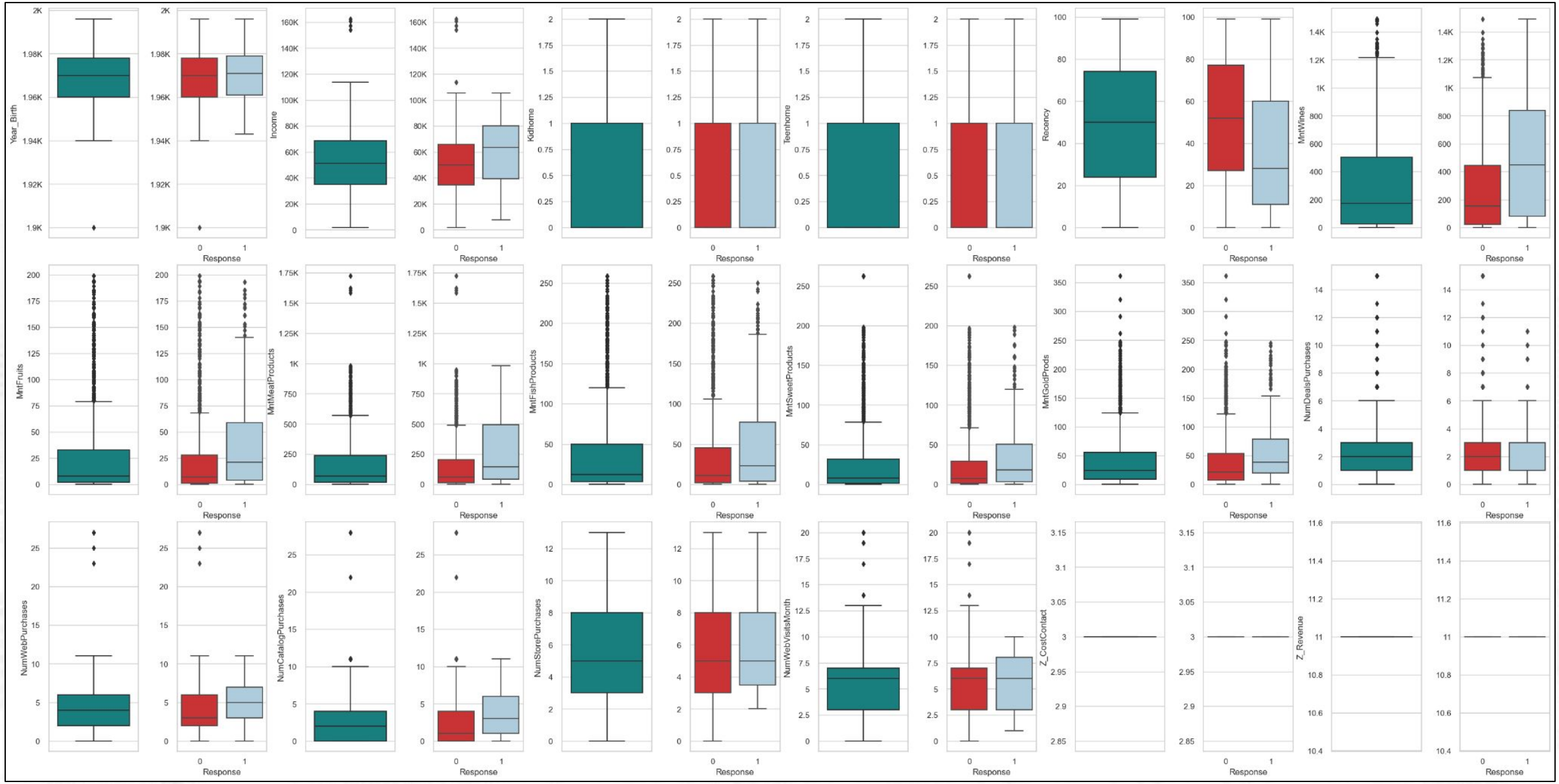
Pada proses handling missing values terdapat **missing values** sebanyak **24 rows (1,07%)** pada kolom **Income**. Dikarenakan keterbatasan data yang ada, pada prosesnya kita **tidak melakukan penghapusan baris (Drop Rows)**, melainkan **dilakukan proses Imputation**. Proses handling missing values yang dilakukan menggunakan **Imputation using MICE with LightGBM**. Imputasi **MICE** dapat **lebih efisien** ketika menggunakan **miceforest** karena diharapkan **kinerjanya jauh lebih baik** saat **mengimplementasikan algortima lightgbm di backend** untuk melakukan imputasi. **LightGBM** dikenal dengan akurasi prediksi yang tinggi ketika digabungkan dengan algortima MICE sehingga menjadikannya algortima yang kuat untuk imputasi.

# Handling Outliers

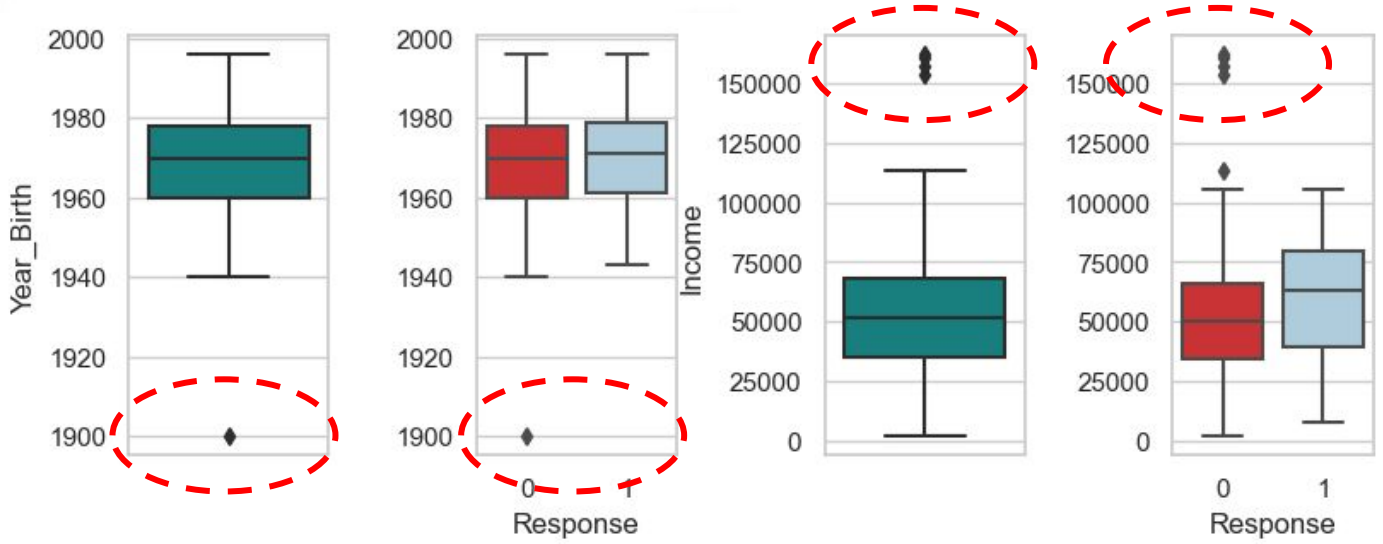
**Jumlah baris: 1680**
**Outlier All Data : 536**
**Not Outlier All Data : 1144**

| | Column Name | is Outlier | Lower Limit | Upper Limit | Outlier | No Outlier |
|---|---|---|---|---|---|---|
| 0 | Year_Birth | True | 1933 | 2005 | 1 | 1679 |
| 1 | Income | True | -15279.250 | 118574.750 | 4 | 1676 |
| 2 | Kidhome | False | -1.500 | 2.500 | 0 | 1680 |
| 3 | Teenhome | False | -1.500 | 2.500 | 0 | 1680 |
| 4 | Recency | False | -51 | 149 | 0 | 1680 |
| 5 | MntWines | True | -693.750 | 1220.250 | 26 | 1654 |
| 6 | MntFruits | True | -44.500 | 79.500 | 179 | 1501 |
| 7 | MntMeatProducts | True | -315.875 | 569.125 | 122 | 1558 |
| 8 | MntFishProducts | True | -67.500 | 120.500 | 171 | 1509 |
| 9 | MntSweetProducts | True | -45.500 | 78.500 | 192 | 1488 |
| 10 | MntGoldProds | True | -60 | 124 | 168 | 1512 |
| 11 | NumDealsPurchases | True | -2 | 6 | 61 | 1619 |
| 12 | NumWebPurchases | True | -4 | 12 | 4 | 1676 |
| 13 | NumCatalogPurchases | True | -6 | 10 | 18 | 1662 |
| 14 | NumStorePurchases | False | -4.500 | 15.500 | 0 | 1680 |
| 15 | NumWebVisitsMonth | True | -3 | 13 | 8 | 1672 |
| 16 | Z_CostContact | False | 3 | 3 | 0 | 1680 |
| 17 | Z_Revenue | False | 11 | 11 | 0 | 1680 |

# Handling Outliers



# Handling Outliers




- Karena pada kolom **Year_Birth** memiliki nilai min yang sangat jauh di tahun **1893-1900**
- **Income** memiliki nilai max yang sangat tinggi sebesar **$666.666**

Maka akan dilakukan penghapusan rows pada nilai ini agar tidak ada ketimpangan nilai. Ada beberapa metode yang dapat kita lakukan :

- **Handling Oulier**
  - ❏ IQR (Interquartile Range)
  - ❏ Z-Score
- **Manually Trimmed**

**Choice Determination:**

- Untuk kasus saat ini, akan digunakan metode **Manually Trimmed**, agar menghindari penghapusan data yang terlalu banyak jika menggunakan Handling Outlier
- Adapun pada kolom lainnya **selain Year_Birth dan Income** yang terdapat outlier tidak kita handle karena akan melalui proses **Normal Distribution Transformation** nantinya yang akan **mereduksi outliernya.**

# Handling Outliers

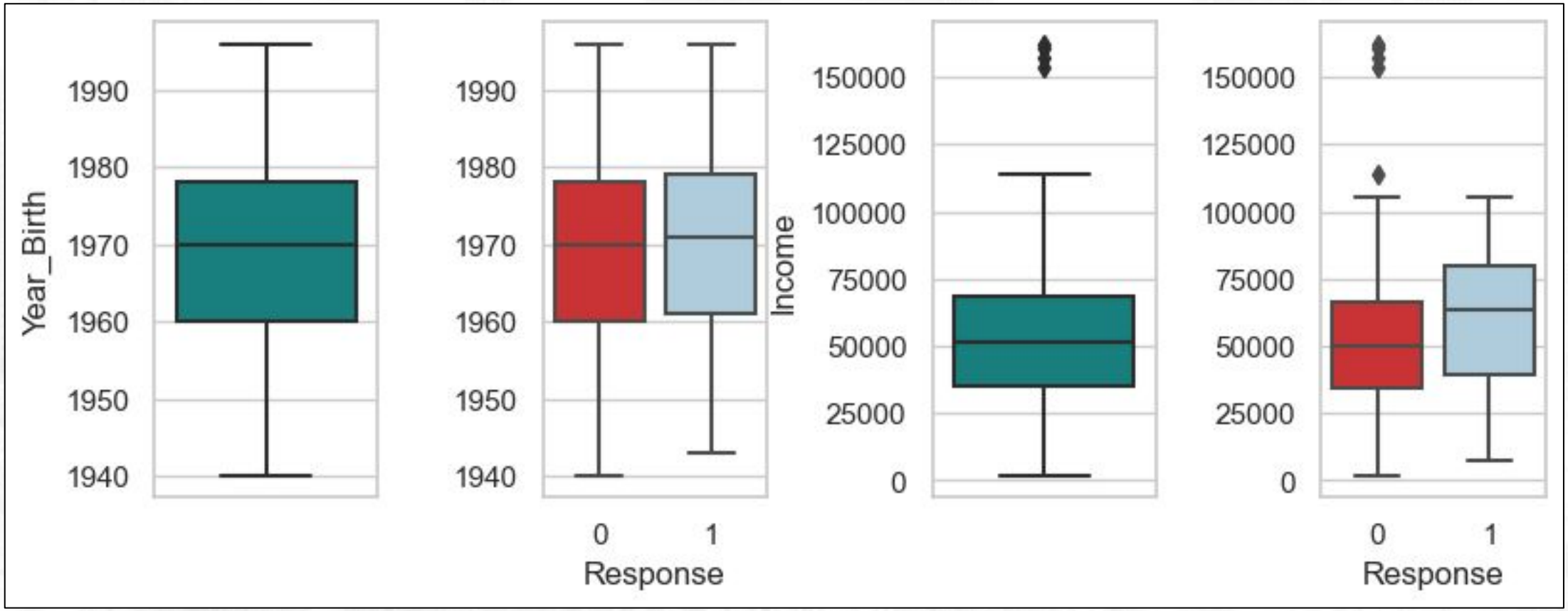
| | Year_Birth | Income |
|---|---|---|
| **count** | 1679 | 1679 |
| **mean** | 1969.202 | 51736.433 |
| **std** | 11.678 | 21476.592 |
| **min** | 1940 | 1730 |
| **25%** | 1960 | 34916 |
| **50%** | 1970 | 51287 |
| **75%** | 1978 | 68407 |
| **max** | 1996 | 162397 |

**Manually Trimmed**

- Kolom Year_Birth, menghapus nilai yang sangat jauh di tahun 1893-1900
- Kolom Income menghapus nilai yang sangat tinggi sebesar $666.666

Rakamin Academy

# Mengecek hasil Trimmed / Drop Outliers



# Handling Outliers

Berdasarkan hasil perhitungan menggunakan **Z-score** dan juga **IQR,** dapat diketahui bahwa jumlah baris yang dihapus dari **Year_Birth dan Income** berdasarkan IQR untuk kolom tidak jauh berbeda dibandingkan dengan Z-score, yaitu :

- **IQR** :
    - Jumlah data sebelum handling outliers : 1680
    - Jumlah data setelah handling outliers (Year_Birth) : 1679
    - Jumlah data setelah handling outliers (Income) : 1675

- **Z_Score** :
    - Jumlah data sebelum handling outliers : 1680
    - Jumlah data setelah handling outliers (Year_Birth) : 1679
    - Jumlah data setelah handling outliers (Income) : 1675

Namun, karena kita ingin meminimalisasi penghapusan data maka untuk proses ini kita memiliki **Manually Trimmed** agar tidak terlalu banyak data yang dihapus, jadi hanya berfokus pada data yang memiliki jauh yang sangat tinggi

- Jumlah data sebelum handling outliers : 1680
- Jumlah data setelah handling outliers (Year_Birth) : 1679
- Jumlah data setelah handling outliers (Income) : 1679

# Feature Engineering / Extraction

Kita akan melakukan Calculation, Extraction, dan Binning features :

1. Kolom **Umur / Age Customer**

   Berdasarkan data diketahui basis tahunnya : **SAS Institute, 2014**. Jadi umur customer didapatkan dari tahun 2014 dikurangi dengan tahun kelahirannya.

```
# currentYear = datetime.now().year
currentYear = 2014 # based on data
df_all['Age'] = currentYear - df_all['Year_Birth']
```

2. Kolom **Age Group**

   Dari hasil umur customer akan disederhanakan menjadi 3 group : **Young Adult < 30, Adult 30-45 Tahun dan Senior Adult > 45 tahun**.

```
def age_group(x):
    if x > 45:
        grup = 'Senior Adult'
    elif x > 30:
        grup = 'Adult'
    else:
        grup = 'Young Adult'

    return grup

df_all['Age_group'] = df_all["Age"].apply(lambda x: age_group(x))
```

# Feature Engineering / Extraction

3. Kolom **Has Child**

   Menggabungkan **Kidhome** dan **Teenhome** menjadi feature **Has_child,** yang mana hasil penjumlahannya yang **memiliki anak minimal 1**.

```
df_all['Has_child'] = np.where(df_all["Kidhome"]+df_all["Teenhome"] > 0, 1, 0)
```

4. Kolom **Dependents**

   Jumlah tanggungan dari customer, dari **penjumlahan Kidhome dan Teenhome**.

```
df_all['Dependents'] = df_all['Kidhome'] + df_all['Teenhome']
```

5. Kolom **Lifetime**

   Sudah berapa bulan customer sejak pembelian pertama di supermarket.

```
df_all['Lifetime'] = (2014 - df_all["Dt_Customer"].dt.year)*12 + df_all["Dt_Customer"].dt.month
```

# Feature Engineering / Extraction

6. Kolom **Spending**

   Jumlah pembelian tiap customer pada keseluruhan product.

```
df['Spending']=df['MntWines']+ df['MntFruits']+ df['MntMeatProducts']+ df['MntFishProducts']+ df['MntSweetProducts']+ df['MntGoldProds']
```

7. Kolom **Primer and Tersier product**

   Jumlah pembelian tiap customer pada kelompok primer dan tersier product.

```
df['Primer_purchase'] = df['MntFruits']+df['MntMeatProducts']+df['MntFishProducts']
df['Tersier_purchase'] = df['MntWines']+df['MntSweetProducts']+df["MntGoldProds"]
```

8. Kolom **Total of Purchases**

   Jumlah pembelian tiap customer pada keseluruhan metode pembelian.

```
df['Total_Purchases'] = df['NumDealsPurchases'] + df['NumWebPurchases'] + df['NumCatalogPurchases'] + df['NumStorePurchases']
```

9. Kolom **Total_Cmp**

   Jumlah tiap customer merespon ke 5 campaign yang dilaksanakan (AcceptedCmp 1 - 5).

```
df['Total_Cmp']=df['AcceptedCmp1'].astype("int64")+ df['AcceptedCmp2'].astype("int64")+ df['AcceptedCmp3'].astype("int64")+ \
    df['AcceptedCmp4'].astype("int64")+ df['AcceptedCmp5'].astype("int64")
```

# Feature Engineering / Extraction

10. Kolom **Ever_Accept**

    Apakah Customer pernah minimal sekali menerima campaign atau tidak pernah sama sekali.

```
df_all['Ever_Accept']=df_all['Total_Cmp'].apply(lambda x : 1 if x >= 1 else 0)
```

11. Kolom **Total Revenue**

    Jumlah Campaign yang diresponse/accept (Campaign 1-5) dikali dengan revenue = 11.

```
df_all['Total_revenue'] = (df_all['Total_Cmp']) * df_all['Z_Revenue']
```

12. Kolom **Income Segmentation**

    Terbagi menjadi 4 segment : None -> Missing values, High -> >= q3(68468), Medium -> q1(35335) - q3(68468) dan Low -> < q1(35335).

```
Q1 = df_all["Income"].quantile(.25)
print(Q1)
Q3 = df_all["Income"].quantile(.75)
print(Q3)
```

```
def income_sgmt(x):
    if (x is None) or (type(x) not in [int, float]) :
        segment = "None"
    else:
        if x >= Q3:
            segment = "High"
        elif x < Q3 and x >= Q1:
            segment = "Medium"
        else:
            segment = "Low"
    return segment
```

```
df_all["Income_sgmt"] = df_all["Income"].fillna("None")
df_all["Income_sgmt"] = df_all["Income_sgmt"].apply(lambda x: income_sgmt(x))
```

# Feature Engineering / Extraction

13. Kolom **Conversion Rate Web**

    Perbandingan Total Purchases dengan Jumlah Pengunjung Website.

```
df_all['Conversion_rate_web'] = np.round(df_all['Total_Purchases'] / df_all['NumWebVisitsMonth'], 2)
df_all['Conversion_rate_web'].fillna(0, inplace=True)
df_all['Conversion_rate_web'].replace([np.inf, -np.inf], 0, inplace=True)
```

14. Kolom **Month Joined**

    Membuat kolom extraction month dari tanggal Customer pertama kali berbelanja.

```
df_all['Month_joined'] = df_all['Dt_Customer'].dt.month
```

15. Kolom **Recency_sgmt**

    Perkiraan pembagian dengan rentang 19 Hari : 4 score -> setengah bulan, 3 score -> 1 bulan, 2 score -> 1 setengah bulan, 1 score -> 2 bulan dan 0 score -> 3 bulan.

| | Recency | | |
|---|---|---|---|
| | min | max | count |
| Recency_sgmt | | | |
| 0 | 80 | 99 | 446 |
| 1 | 60 | 79 | 437 |
| 2 | 40 | 59 | 450 |
| 3 | 20 | 39 | 447 |
| 4 | 0 | 19 | 456 |

```
divided = {5: 19, 4: 39, 3: 59, 2: 79}

def RScore(x,d):
    if x <= d[5]:
        return 4
    elif x <= d[4]:
        return 3
    elif x <= d[3]:
        return 2
    elif x <= d[2]:
        return 1
    else:
        return 0

df_all['Recency_sgmt'] = df_all['Recency'].apply(lambda x: RScore(x, divided))
```

# Feature Engineering / Extraction

## Hasil Feature Engineering / Extraction

| | ID | Age | Age_group | Has_child | Dependents | Lifetime | Spending | Primer_purchase | Tersier_purchase |
|---|---|---|---|---|---|---|---|---|---|
| **413** | 6504 | 39 | Adult | 1 | 1 | 23 | 78 | 42 | 36 |
| **581** | 5756 | 31 | Adult | 1 | 1 | 4 | 55 | 7 | 48 |
| **1225** | 8210 | 39 | Adult | 1 | 1 | 20 | 405 | 24 | 381 |
| **1324** | 839 | 39 | Adult | 1 | 1 | 21 | 170 | 54 | 116 |
| **1000** | 5527 | 27 | Young Adult | 1 | 1 | 14 | 58 | 26 | 32 |

| | ID | Total_Purchases | Total_Cmp | Ever_Accept | Total_revenue | Income_sgmt | Conversion_rate_web | Month_joined | Recency_sgmt |
|---|---|---|---|---|---|---|---|---|---|
| **413** | 6504 | 9 | 1 | 1 | 11 | Low | 1.80 | 11 | 3 |
| **581** | 5756 | 7 | 0 | 0 | 0 | Medium | 1.17 | 4 | 1 |
| **1225** | 8210 | 18 | 0 | 0 | 0 | Medium | 4.50 | 8 | 1 |
| **1324** | 839 | 9 | 0 | 0 | 0 | Medium | 1.80 | 9 | 2 |
| **1000** | 5527 | 7 | 0 | 0 | 0 | Low | 0.88 | 2 | 3 |

# Feature Engineering / Extraction

## Hasil Keseluruhan Feature

**Categorical (String)**

- Education - Basic, Graduation, Master, PhD
- Marital_Status - Single, Married, Divorced
- Age_group - Young Adult, Adult, Senior Adult
- Income_sgmt - High, Medium, Low

**Categorical (Int)**

- ID
- Kidhome - 0, 1, 2
- Teenhome - 0, 1, 2
- AcceptedCmp1 - 0, 1
- AcceptedCmp2 - 0, 1
- AcceptedCmp3 - 0, 1
- AcceptedCmp4 - 0, 1
- AcceptedCmp5 - 0, 1
- Ever_Accept - 0, 1
- Complain - 0, 1
- Response - 0, 1
- Has_child - 0, 1
- Recency_sgmt - 0, 1, 2, 3, 4

**Numericals**

- Year_Birth = 1940 - 1996
- Income = 1730.0 - 162397.0
- Kidhome = 0 - 2
- Teenhome = 0 - 2
- Recency = 0 - 99
- Age = 18 - 74
- Dependents = 0 - 3
- Lifetime = 1 - 36
- Spending = 5 - 2525
- Primer_purchase = 1 - 1727
- Tersier_purchase = 3 - 1689
- Total_Purchases = 0 - 44
- NumWebVisitsMonth = 0 - 20
- Conversion_rate_web = 0.0 - 43.0
- Total_Cmp = 0 - 4
- Total_revenue = 0 - 44
- Month_joined = 1 - 12

**Numericals (one)**

- Z_CostContact = 3
- Z_Revenue = 11

**Numericals (Product)**

- MntWines = 0 - 1493
- MntFruits = 0 - 199
- MntMeatProducts = 0 - 1725
- MntFishProducts = 0 - 259
- MntSweetProducts = 0 - 263
- MntGoldProds = 0 - 362

**Numericals (Purchases)**

- NumDealsPurchases = 0 - 15
- NumWebPurchases = 0 - 27
- NumCatalogPurchases = 0 - 28
- NumStorePurchases = 0 - 13

**Timestamp**

- Dt_Customer = 2012-07-30 - 2014-06-29

# Feature Transformation (Numeric)

Mengecek Skewness di tiap kolom untuk menentukan jenis transformation

| | Column Name | Skewness | Kurtosis | Type of Distribution |
|---|---|---|---|---|
| 0 | Kidhome | 0.624 | -0.794 | Bimodal Distribution |
| 1 | Teenhome | 0.442 | -0.918 | Bimodal Distribution |
| 2 | Conversion_rate_web | 2.359 | 6.967 | Highly Positively Skewed |
| 3 | MntFishProducts | 1.921 | 3.072 | Highly Positively Skewed |
| 4 | MntFruits | 2.135 | 4.236 | Highly Positively Skewed |
| 5 | MntGoldProds | 1.925 | 3.755 | Highly Positively Skewed |
| 6 | MntMeatProducts | 2.079 | 5.503 | Highly Positively Skewed |
| 7 | MntSweetProducts | 2.184 | 4.713 | Highly Positively Skewed |
| 8 | MntWines | 1.179 | 0.632 | Highly Positively Skewed |
| 9 | NumCatalogPurchases | 1.891 | 8.297 | Highly Positively Skewed |
| 10 | NumDealsPurchases | 2.528 | 9.902 | Highly Positively Skewed |
| 11 | NumWebPurchases | 1.549 | 6.927 | Highly Positively Skewed |
| 12 | Primer_purchase | 1.589 | 2.209 | Highly Positively Skewed |
| 13 | Total_Cmp | 2.733 | 8.033 | Highly Positively Skewed |
| 14 | Total_revenue | 2.733 | 8.033 | Highly Positively Skewed |
| 15 | Dependents | 0.425 | -0.251 | Moderately Normal Distribution (Symmetric) |
| 16 | Income | 0.264 | 0.381 | Moderately Normal Distribution (Symmetric) |
| 17 | NumWebVisitsMonth | 0.351 | 2.385 | Moderately Normal Distribution (Symmetric) |
| 18 | Total_Purchases | 0.263 | -0.837 | Moderately Normal Distribution (Symmetric) |
| 19 | NumStorePurchases | 0.697 | -0.609 | Moderately Positively Skewed |
| 20 | Spending | 0.839 | -0.402 | Moderately Positively Skewed |
| 21 | Tersier_purchase | 0.966 | 0.039 | Moderately Positively Skewed |
| 22 | Age | 0.116 | -0.749 | Normal Distribution (Symmetric) |
| 23 | Lifetime | 0.038 | -1.038 | Normal Distribution (Symmetric) |
| 24 | Month_joined | -0.026 | -1.282 | Normal Distribution (Symmetric) |
| 25 | Recency | -0.004 | -1.206 | Normal Distribution (Symmetric) |
| 26 | Year_Birth | -0.116 | -0.749 | Normal Distribution (Symmetric) |
| 27 | Z_CostContact | NaN | NaN | Uniform Distribution |
| 28 | Z_Revenue | NaN | NaN | Uniform Distribution |

# Feature Transformation (Numeric)

**Scaling and Converting to a Normal Distribution :**

- Log Transformation
- Box-Cox Transformation
- Yeo-Johnson Transformation

**Column yang akan di transform :**

- ➔ Conversion_rate_web
- ➔ MntFishProducts
- ➔ MntFruits
- ➔ MntGoldProds
- ➔ MntMeatProducts
- ➔ MntSweetProducts
- ➔ MntWines
- ➔ NumCatalogPurchases
- ➔ NumDealsPurchases
- ➔ NumStorePurchases
- ➔ NumWebPurchases
- ➔ Primer_purchase
- ➔ Spending
- ➔ Tersier_purchase
- ➔ Total_revenue

**Just Scaling**

- Normalization
- Standardization

**Column yang akan di transform**

- ➔ Age
- ➔ Income
- ➔ Lifetime
- ➔ Month_joined
- ➔ NumWebVisitsMonth
- ➔ Recency
- ➔ Total_Purchases
- ➔ Year_Birth

**Kolom yang *tidak perlu melakukan Transformasi* karena rentang nilai yang masih wajar sebagai berikut :**

- ➔ Kidhome
- ➔ Teenhome
- ➔ Dependents
- ➔ Total_Cmp

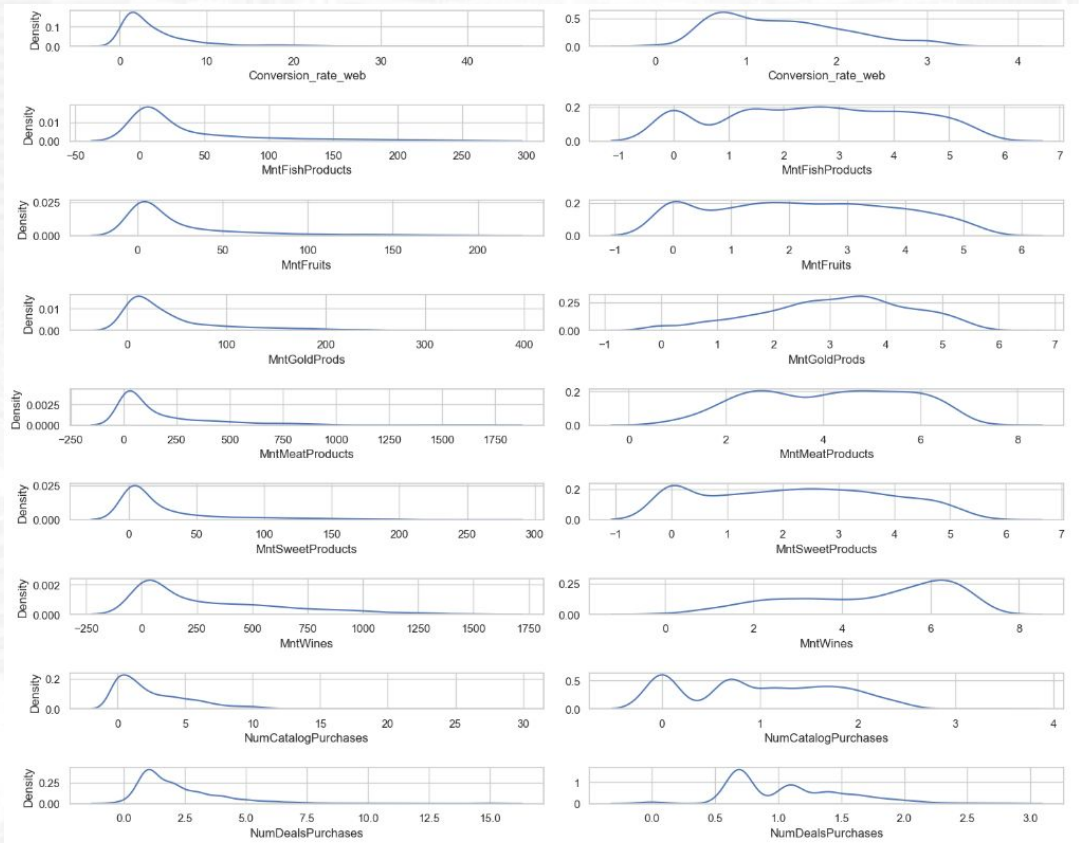
Choice Determination:

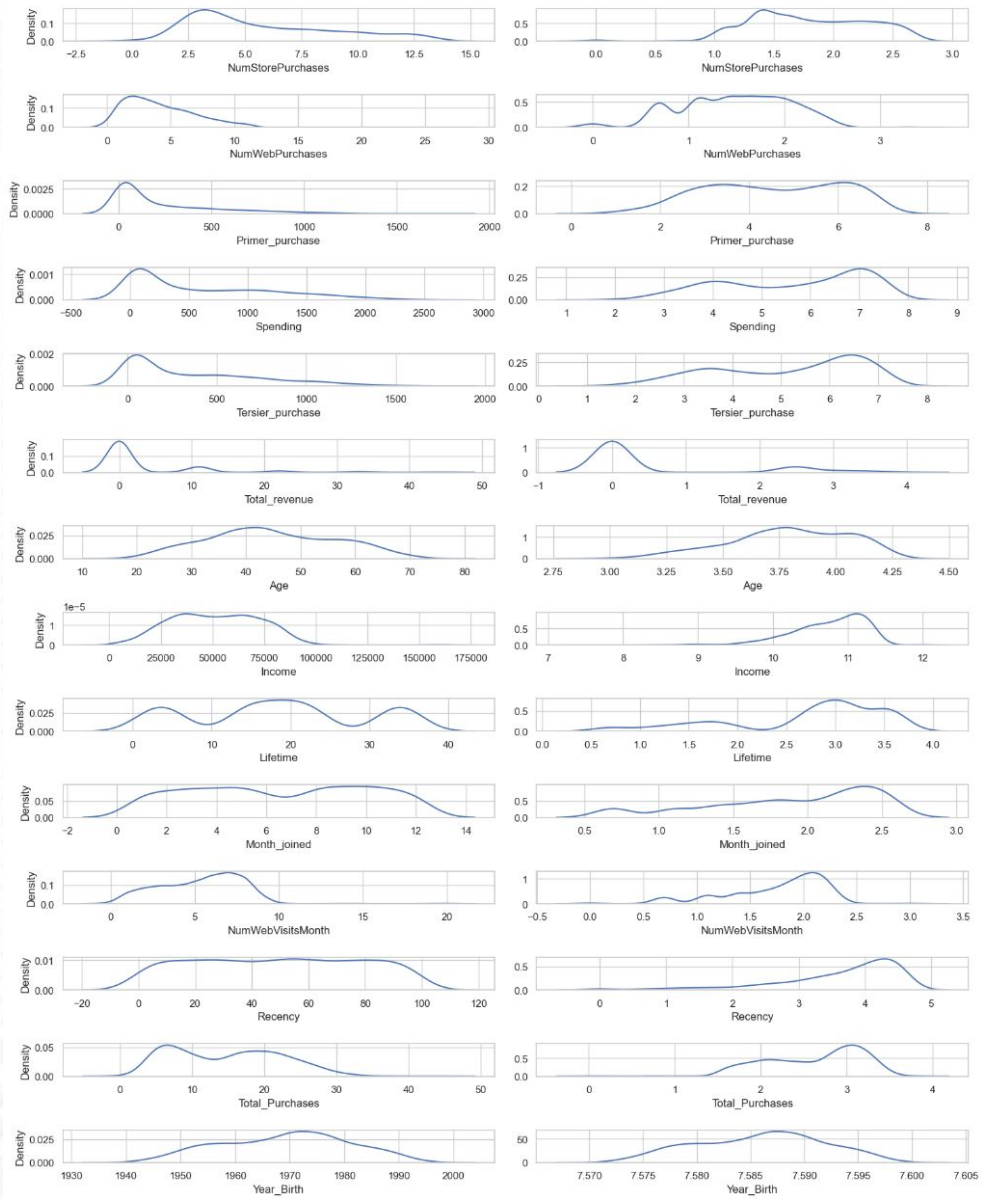
- Pada proses **Feature Transformation / Scaling** ini kita menggunakan **Yeo-Johnson Transformation** pada kolom-kolom yang masih memiliki skala yang besar, karena dari hasilnya kita bisa melihat hasil bentuk curve yang lebih Normal Distribusi. Dan sangat cocok untuk penggunaan Algoritma berbasis tree.
- dari proses evaluasi model, didapatkan penggunaan transformasi yang hanya menggunakan **Yeo-Johnson Transformation** pada kolom-kolom yang masih memiliki skala yang besar, memberikan hasil yang lumayan akurat maka untuk kolom skew maupun normal akan dilakukan transformasi yang sama.
- Sehingga untuk **Normalization dan Standardization** tidak akan digunakan

# Feature Transformation (Numeric)

## Log Transformation

```
fig, ax = plt.subplots(len(log_cols),2,figsize=(15,30))
for i in range(0,len(log_cols)):
    kde1 = sns.kdeplot(df_train[log_cols[i]], ax=ax[i][0])
    kde2 = sns.kdeplot(np.log(df_train[log_cols[i]]+1), ax=ax[i][1])
    kde2.set_ylabel(None)
    plt.tight_layout()
```

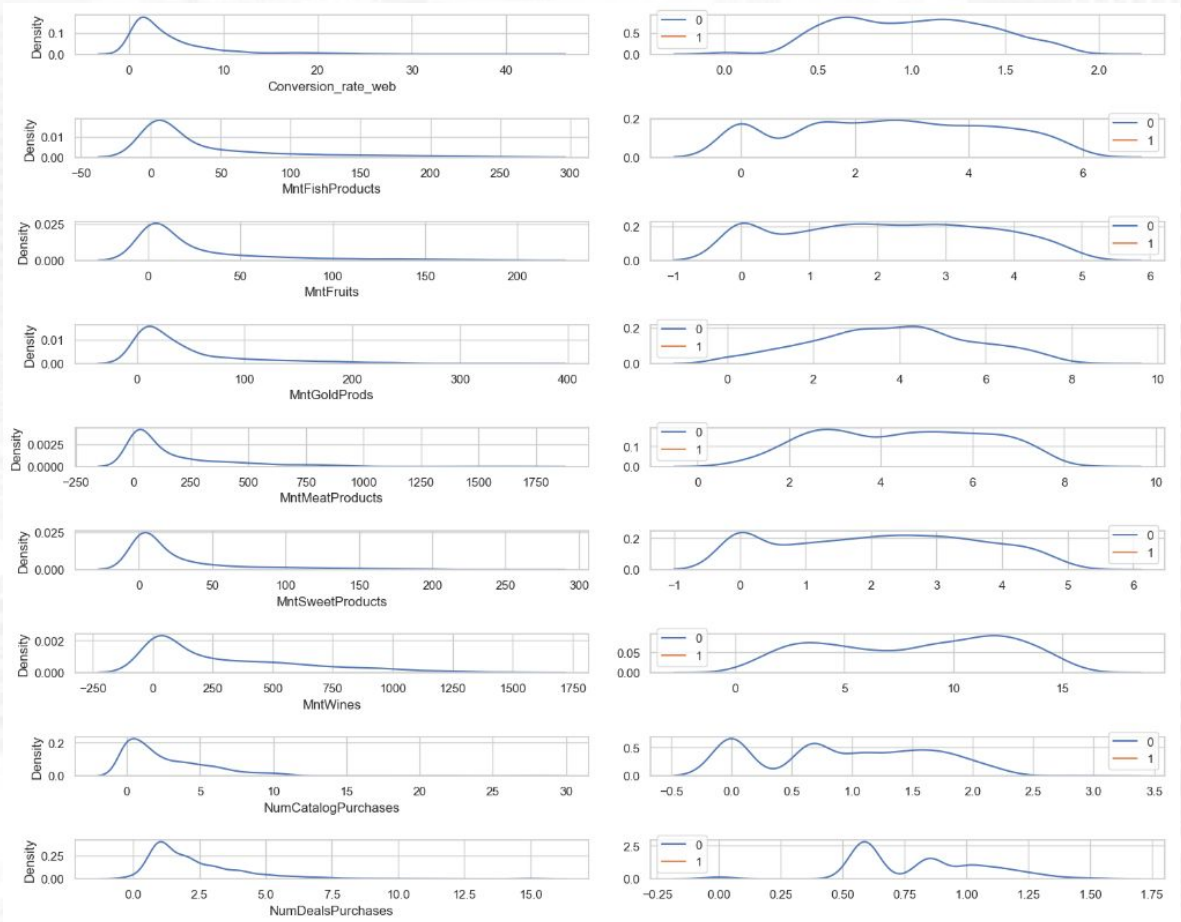


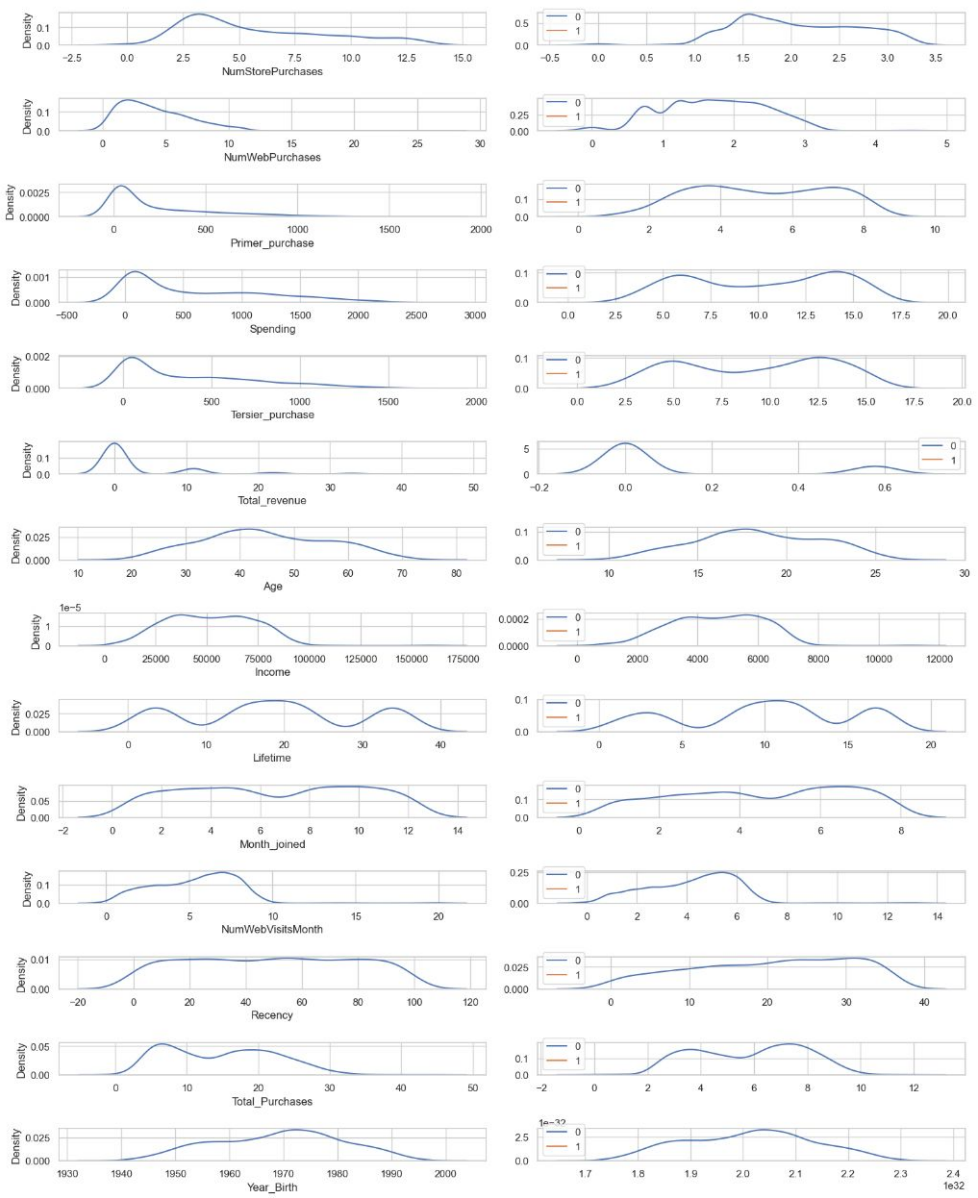



# Feature Transformation (Numeric)

## Box-Cox Transformation (with Scipy)

```
from scipy.stats import boxcox
fig, ax = plt.subplots(len(log_cols),2,figsize=(15,30))
for i in range(0,len(log_cols)):
    kde1 = sns.kdeplot(df_train[log_cols[i]], ax=ax[i][0])
    kde2 = sns.kdeplot(boxcox(df_train[log_cols[i]]+1), warn_singular=False, ax=ax[i][1])
    kde2.set_ylabel(None)
    plt.tight_layout()
```

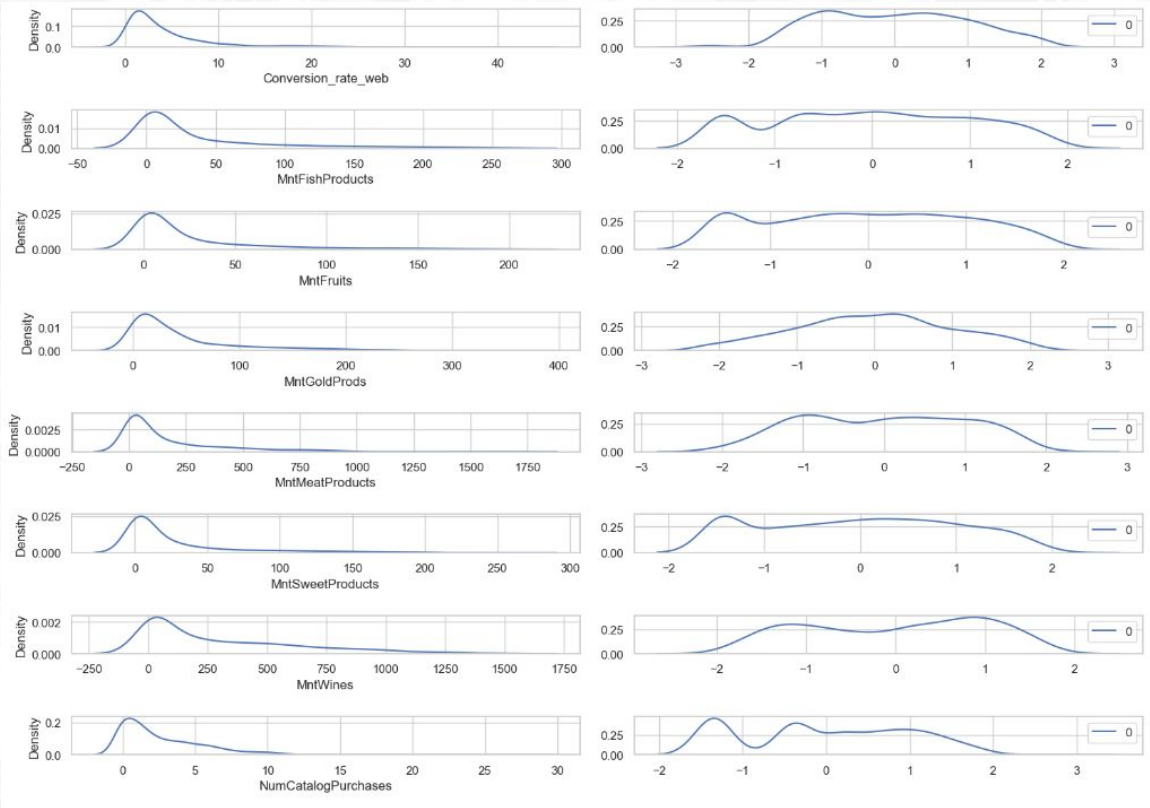


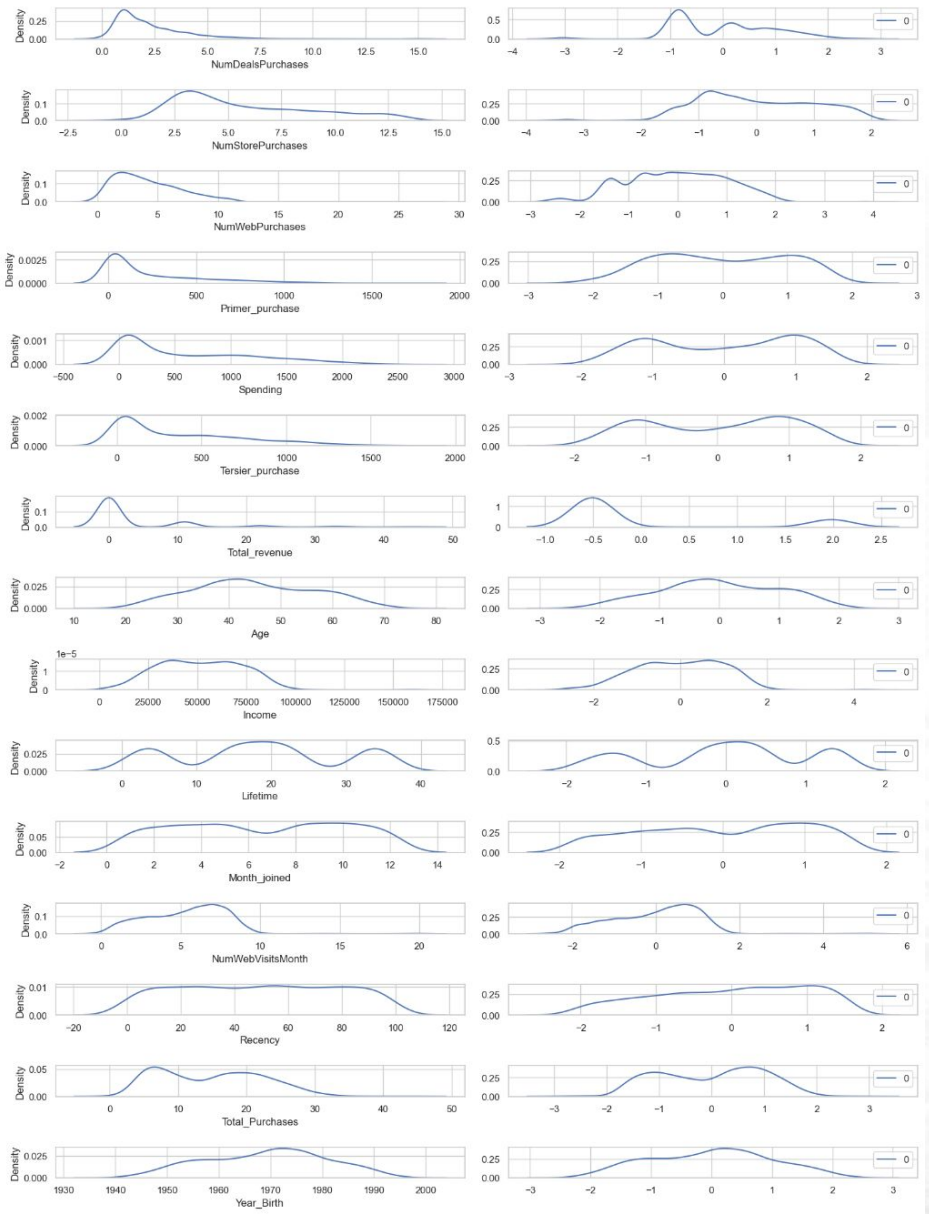



# Feature Transformation (Numeric)

## Box-Cox Transformation (with Sklearn)

```
from sklearn.preprocessing import PowerTransformer

fig, ax = plt.subplots(len(log_cols),2,figsize=(15,30))
for i in range(0,len(log_cols)):
    pt = PowerTransformer(method='box-cox')
    data = pt.fit_transform(df_train[[log_cols[i]]]+1)
    kde1 = sns.kdeplot(df_train[log_cols[i]], ax=ax[i][0])
    kde2 = sns.kdeplot(data, ax=ax[i][1])
    kde2.set_ylabel(None)
    plt.tight_layout()
```



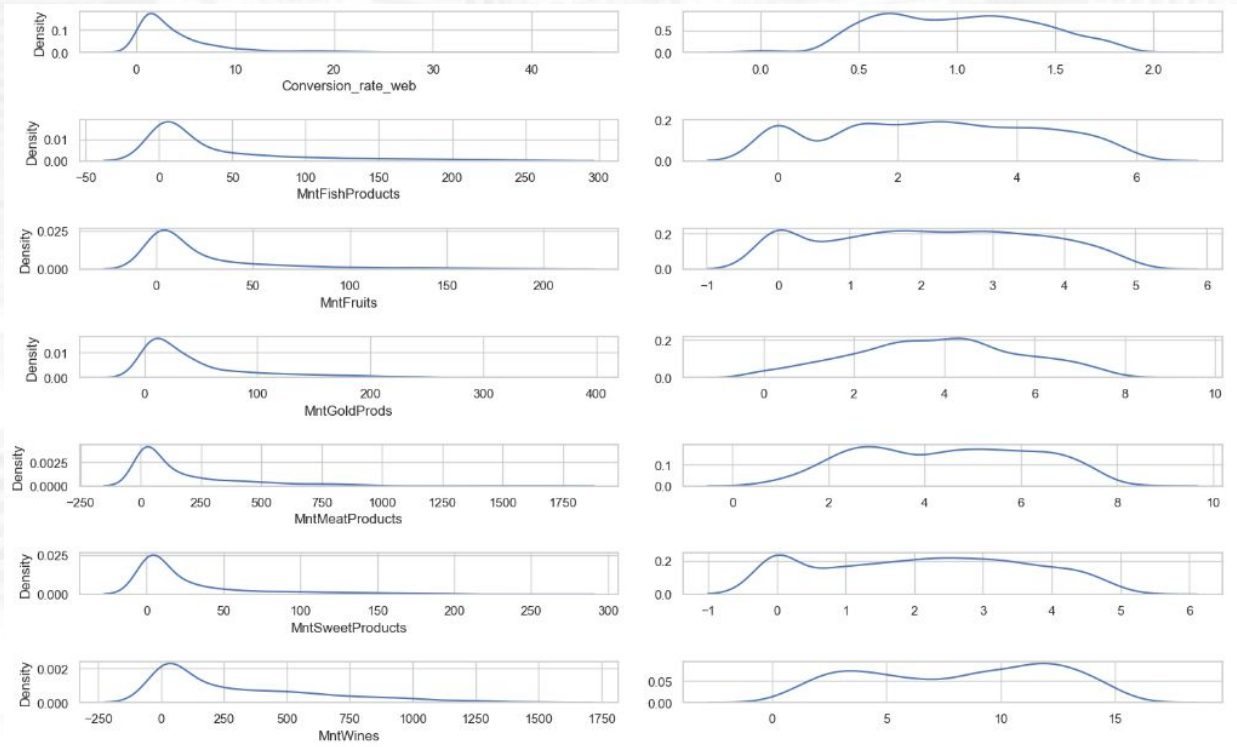



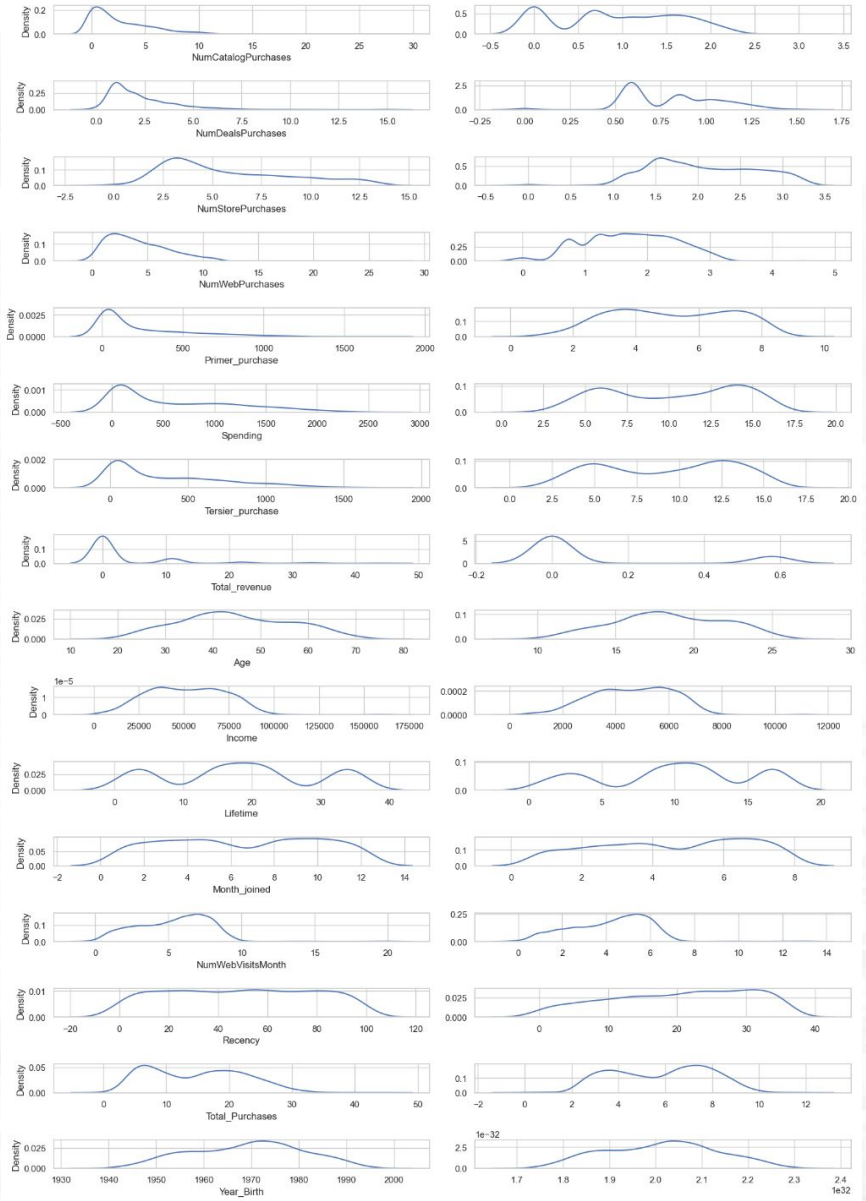
# Feature Transformation (Numeric)

## Yeo-Johnson Transformation (with Scipy)

Unlike the Box-Cox transform, it does not require the values for each input variable to be strictly positive. It supports zero values and negative values.
This means we can apply it to our dataset without scaling it first.

```
from scipy.stats import yeojohnson

fig, ax = plt.subplots(len(log_cols),2,figsize=(15,30))
for i in range(0,len(log_cols)):
    data, fitted_lambda = yeojohnson(df_train[log_cols[i]],lmbda=None)
    kde1 = sns.kdeplot(df_train[log_cols[i]], ax=ax[i][0])
    kde2 = sns.kdeplot(data, ax=ax[i][1])
    kde2.set_ylabel(None)
    plt.tight_layout()
```

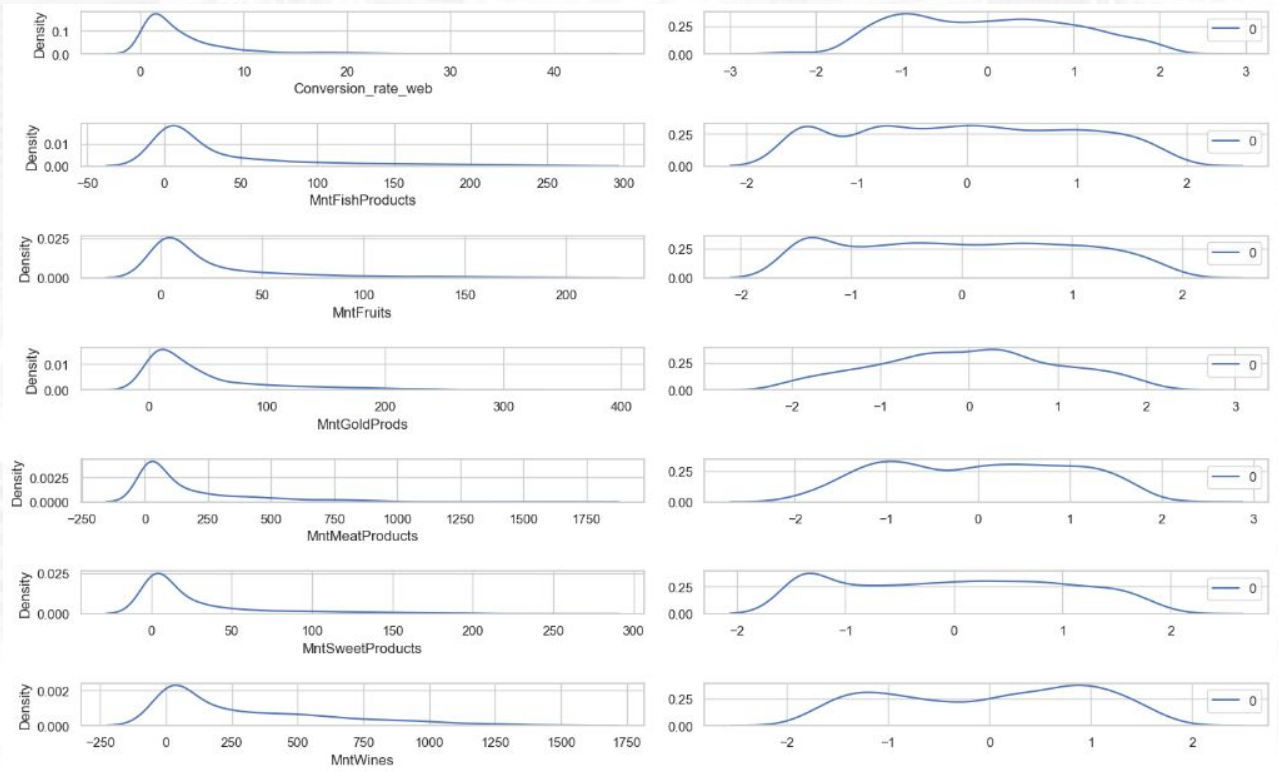


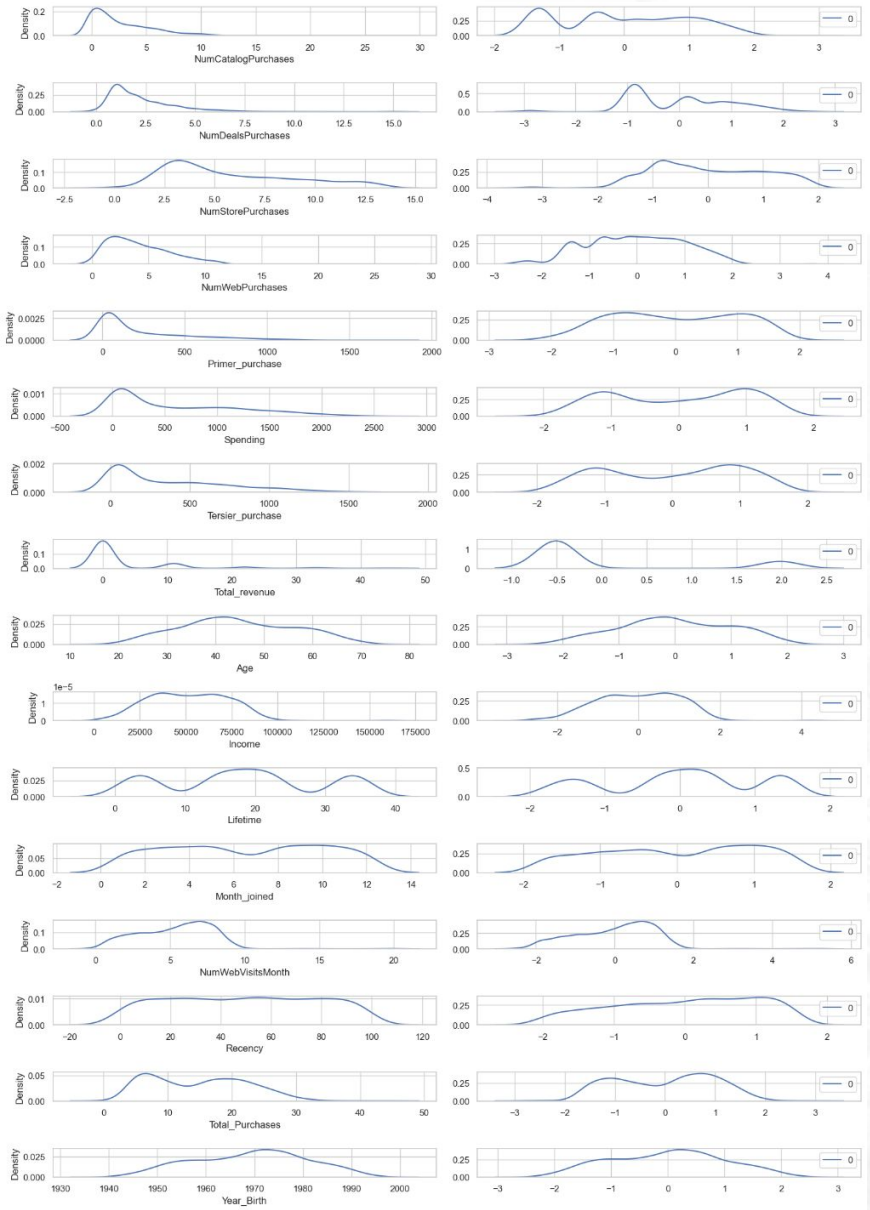



# Feature Transformation (Numeric)

## Yeo-Johnson Transformation (with Sklearn)

```
from sklearn.preprocessing import PowerTransformer

fig, ax = plt.subplots(len(log_cols),2,figsize=(15,30))
for i in range(0,len(log_cols)):
    pt = PowerTransformer(method='yeo-johnson')
    data = pt.fit_transform(df_train[[log_cols[i]]]+1)
    kde1 = sns.kdeplot(df_train[log_cols[i]], ax=ax[i][0])
    kde2 = sns.kdeplot(data, ax=ax[i][1])
    kde2.set_ylabel(None)
    plt.tight_layout()
```



# Feature Transformation (Numeric)

## Yeo-Johnson Transformation (with Sklearn)

| | Conversion_rate_web | MntFishProducts | MntFruits | MntGoldProds | MntMeatProducts | MntSweetProducts | MntWines | NumCatalogPurchases | NumDealsPurchases | NumStorePurchases |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 1679 | 1679 | 1679 | 1679 | 1679 | 1679 | 1679 | 1679 | 1679 | 1679 |
| mean | -0.000 | -0.000 | 0.000 | -0.000 | 0.000 | 0.000 | -0.000 | 0.000 | 0.000 | -0.000 |
| std | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| min | -2.568 | -1.520 | -1.472 | -2.117 | -2.122 | -1.435 | -1.981 | -1.340 | -3.040 | -3.302 |
| 25% | -0.843 | -0.710 | -0.728 | -0.676 | -0.850 | -0.971 | -0.913 | -1.340 | -0.837 | -0.854 |
| 50% | -0.022 | -0.005 | -0.009 | 0.011 | 0.033 | 0.002 | 0.125 | 0.158 | 0.148 | -0.045 |
| 75% | 0.762 | 0.831 | 0.829 | 0.676 | 0.851 | 0.806 | 0.878 | 0.808 | 0.738 | 0.810 |
| max | 2.397 | 1.854 | 1.893 | 2.474 | 2.228 | 2.027 | 1.821 | 2.834 | 2.668 | 1.798 |

| NumWebPurchases | Primer_purchase | Spending | Tersier_purchase | Total_revenue | Age | Income | Lifetime | Month_joined | NumWebVisitsMonth | Recency | Total_Purchases | Year_Birth |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1679 | 1679 | 1679 | 1679 | 1679 | 1679 | 1679 | 1679 | 1679 | 1679 | 1679 | 1679 | 1679 |
| 0.000 | 0.000 | -0.000 | -0.000 | 0.000 | 0.000 | 0.000 | -0.000 | 0.000 | -0.000 | -0.000 | 0.000 | -0.000 |
| 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| -2.394 | -2.339 | -2.071 | -1.975 | -0.505 | -2.534 | -2.845 | -1.812 | -1.715 | -2.393 | -2.033 | -2.834 | -2.370 |
| -0.717 | -0.823 | -0.975 | -0.978 | -0.505 | -0.732 | -0.751 | -0.408 | -1.002 | -0.936 | -0.788 | -0.861 | -0.800 |
| 0.177 | -0.046 | 0.126 | 0.124 | -0.505 | -0.027 | 0.034 | 0.052 | -0.090 | 0.297 | 0.128 | 0.141 | 0.042 |
| 0.810 | 0.891 | 0.921 | 0.874 | -0.505 | 0.801 | 0.791 | 0.562 | 0.979 | 0.682 | 0.854 | 0.837 | 0.743 |
| 3.842 | 2.043 | 1.765 | 1.855 | 2.005 | 2.326 | 4.353 | 1.491 | 1.474 | 5.127 | 1.542 | 2.885 | 2.420 |

# Feature Transformation (Numeric)

## Normalization

```
from sklearn.preprocessing import MinMaxScaler
# create a scaler object
scaler = MinMaxScaler()
# fit and transform the data
df[norm_cols] = pd.DataFrame(scaler.fit_transform(df[norm_cols]), columns=df[norm_cols].columns)
```

```
df[norm_cols].describe()
```

| | Age | Income | Lifetime | Month_joined | NumWebVisitsMonth | Recency | Total_Purchases | Year_Birth |
|---|---|---|---|---|---|---|---|---|
| count | 2236.000000 | 2236.000000 | 2236.000000 | 2236.000000 | 2236.000000 | 2236.000000 | 2236.000000 | 2236.000000 |
| mean | 0.483964 | 0.312764 | 0.489522 | 0.496910 | 0.265944 | 0.496124 | 0.338012 | 0.516036 |
| std | 0.208987 | 0.133932 | 0.306805 | 0.317167 | 0.121344 | 0.292498 | 0.174497 | 0.208987 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.339286 | 0.209162 | 0.342857 | 0.181818 | 0.150000 | 0.242424 | 0.181818 | 0.339286 |
| 50% | 0.464286 | 0.308968 | 0.485714 | 0.454545 | 0.300000 | 0.494949 | 0.340909 | 0.535714 |
| 75% | 0.660714 | 0.415382 | 0.657143 | 0.818182 | 0.350000 | 0.747475 | 0.477273 | 0.660714 |
| max | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |

Berdasarkan hasil pengecekan pada beberapa fitur yang telah diproses menggunakan transformation sebelumnya, dapat diketahui bahwa keseluruhan nilai skewnessnya sudah memiliki rentang yang lebih seragam (tidak jauh dan tidak terlalu bervariasi). Sehingga dapat disimpulkan bahwa teknik fitur transformation yang telah kami lakukan sudah valid dan kami.

# Feature Encoding (Categoric)

Melihat feature categorical yang masih memiliki nilai betype string/object

```
#cek unique value
for x in cat_str:
    unq = list(df[x].unique())
    sorted(unq)

    print (f'===== {x} =====')
    if len(unq) >= 10:
        unq = list(unq[:10])+['.....']
        print (f'{unq}')
    else:
        print (f'{unq}')
    print()
```

```
===== Education =====
['Graduation', 'PhD', 'Master', 'Basic']

===== Marital_Status =====
['Single', 'Married', 'Divorced']

===== Age_group =====
['Senior Adult', 'Young Adult', 'Adult']

===== Income_sgmt =====
['Medium', 'High', 'Low']
```

```
cat_str
```

```
['Education', 'Marital_Status', 'Age_group', 'Income_sgmt']
```

Berdasarkan hasil analisis, berikut beberapa encoding yang dilakukan:

**1. Label Encoding**

- **Label Encoder**
- **Manually Mapped**

Adapun daftar column yang akan diproses :

- **Education** => Basic (0), Graduation (1), Master (2), PhD (3)
- **Age_group** => Young Adult (0), Adult (1), Senior Adult (2)
- **Income_sgmt** => Low (0), Medium (1), High (2)

**2. One Hot Encoding**

- **get_dummies**
- **OneHotEncoder**

Adapun daftar column yang akan diproses :

- **Marital_Status** => Single, Married, Divorced

**Choice Determination**

- Pada proses **Label Encoding** ini kita menggunakan **Manually Mapped**, karena kita bisa menentukan secara fleksibel urutan/order dari categorical feature
- Pada proses **One Hot Encoding** ini kita menggunakan **OneHotEncoder**, karena hasil encodingnya lebih rapi dan lebih mudah untuk dilakukan adjust

# Feature Encoding (Categoric)

Rakamin Academy

## 1. Label Encoding

a) Menggunakan **Label Encoder**

```
# from sklearn.preprocessing import LabelEncoder

# cat = cat_str.copy()
# cat.remove("Marital_Status")

# le = LabelEncoder()

# for i in cat_str:
#     le.fit(df_train[i])
#     df_train[i] = le.transform(df_train[i])
#     df_test[i] = le.transform(df_test[i])
#     print(le.classes_)
```

b) Menggunakan Metode **Mapping**

```
map_edu = {
    'Basic' : 0,
    'Graduation' : 1,
    'Master' : 2,
    'PhD' : 3
}

df_train['Education'] = df_train['Education'].map(map_edu)
print("Education (Train) = ", np.sort(df_train['Education'].unique()))

df_test['Education'] = df_test['Education'].map(map_edu)
print("Education (Test) = ", np.sort(df_test['Education'].unique()))
```

```
Education (Train) =  [0 1 2 3]
Education (Test) =  [0 1 2 3]
```

```
map_age = {
    'Young Adult' : 0,
    'Adult' : 1,
    'Senior Adult' : 2
}

df_train['Age_group'] = df_train['Age_group'].map(map_age)
print("Age_group (Train) = ", np.sort(df_train['Age_group'].unique()))

df_test['Age_group'] = df_test['Age_group'].map(map_age)
print("Age_group (Test) = ", np.sort(df_test['Age_group'].unique()))
```

```
Age_group (Train) =  [0 1 2]
Age_group (Test) =  [0 1 2]
```

```
map_income = {
    'Low' : 0,
    'Medium' : 1,
    'High' : 2
}

df_train['Income_sgmt'] = df_train['Income_sgmt'].map(map_income)
print("Income_sgmt (Train) = ", np.sort(df_train['Income_sgmt'].unique()))

df_test['Income_sgmt'] = df_test['Income_sgmt'].map(map_income)
print("Income_sgmt (Test) = ", np.sort(df_test['Income_sgmt'].unique()))
```

```
Income_sgmt (Train) =  [0 1 2]
Income_sgmt (Test) =  [0 1 2]
```

# Feature Encoding (Categoric)

## 2. One Hot Encoding

a) Menggunakan **get_dummies**

```
# df_train["Marital_Status"].unique()
```

```
# # TRAIN
# #Not Auto Drop Columns
# ohe = pd.get_dummies(df_train["Marital_Status"])
# df_train.drop("Marital_Status", axis=1, inplace=True)
# df_train = df_train.join(ohe)

# # cara 2 : Auto drop Column
# # df_train = pd.get_dummies(data=df_train, columns=["Marital_Status"])

# df_train.head()
```

```
# df_test["Marital_Status"].unique()
```

```
# # TEST
# #Not Auto Drop Columns
# ohe = pd.get_dummies(df_test["Marital_Status"])
# df_test.drop("Marital_Status", axis=1, inplace=True)
# df_test = df_test.join(ohe)

# # cara 2 : Auto drop Column
# # df_test = pd.get_dummies(data=df_test, columns=["Marital_Status"])

# df_test.head()
```

b) Menggunakan **OneHotEncoder**

```
from sklearn.preprocessing import OneHotEncoder

oh = OneHotEncoder()
df_ohe_train = pd.DataFrame(
    oh.fit_transform(df_train[["Marital_Status"]]).toarray(),
    columns=list(oh.categories_[0])
)
print(oh.categories_[0])

df_train.drop("Marital_Status", axis=1, inplace=True)

df_train = pd.concat([df_train, df_ohe_train], axis=1)
```

```
['Divorced' 'Married' 'Single']
```

```
from sklearn.preprocessing import OneHotEncoder

df_ohe_test = pd.DataFrame(
    oh.transform(df_test[["Marital_Status"]]).toarray(),
    columns=list(oh.categories_[0])
)

df_test.drop("Marital_Status", axis=1, inplace=True)

df_test = pd.concat([df_test, df_ohe_test], axis=1)
```

## Kesimpulan

Berdasarkan hasil pengecekan pada beberapa fitur yang telah diproses menggunakan encoding sebelumnya, dapat diketahui bahwa keseluruhan nilai telah bertipe numeric sesuai dengan nilai yang kita assign. Sehingga dapat disimpulkan bahwa teknik fitur encoding yang telah kami lakukan sudah valid dan kami.

# Feature Selection

## 1. Drop Unnecessary Features

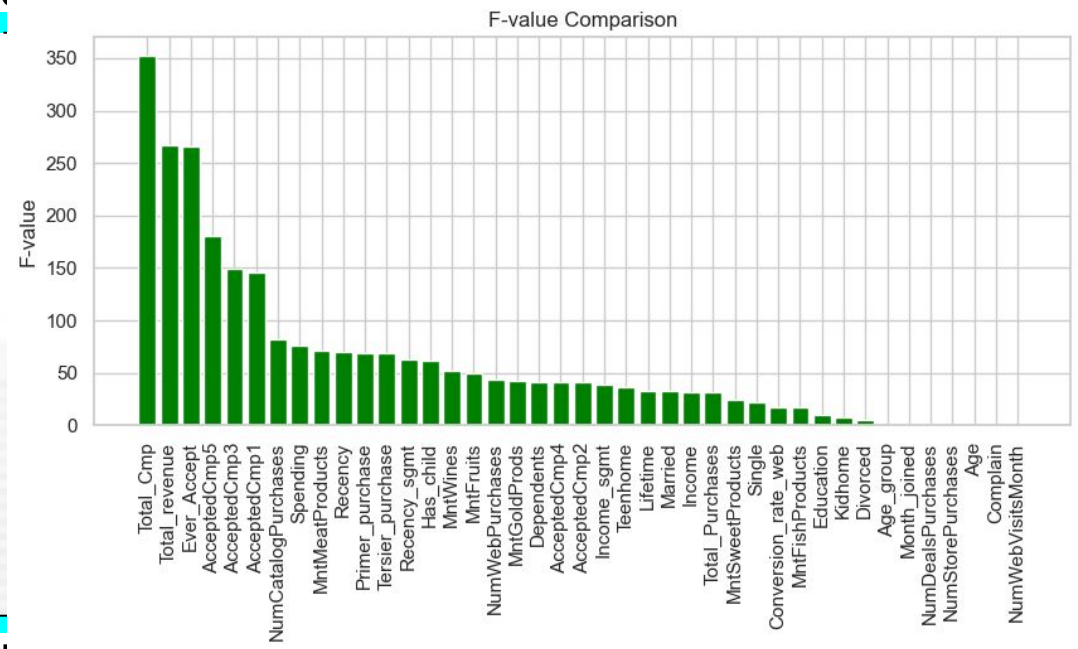
```
df.drop(['ID', 'Year_Birth', 'Dt_Customer', 'Z_CostContact', 'Z_Revenue'], inplace=True, axis=1)
```

- **ID** >> memiliki banyak kategori
- **Year_Birth** >> Feature extraction untuk mengambil data **Age** pada tahun 2014
- **Dt_Customer** >> tidak terlalu mempengaruhi model
- **Z_CostContact** dan **Z_Revenue** >> tidak memberikan informasi yang signifikan terhadap model prediksi

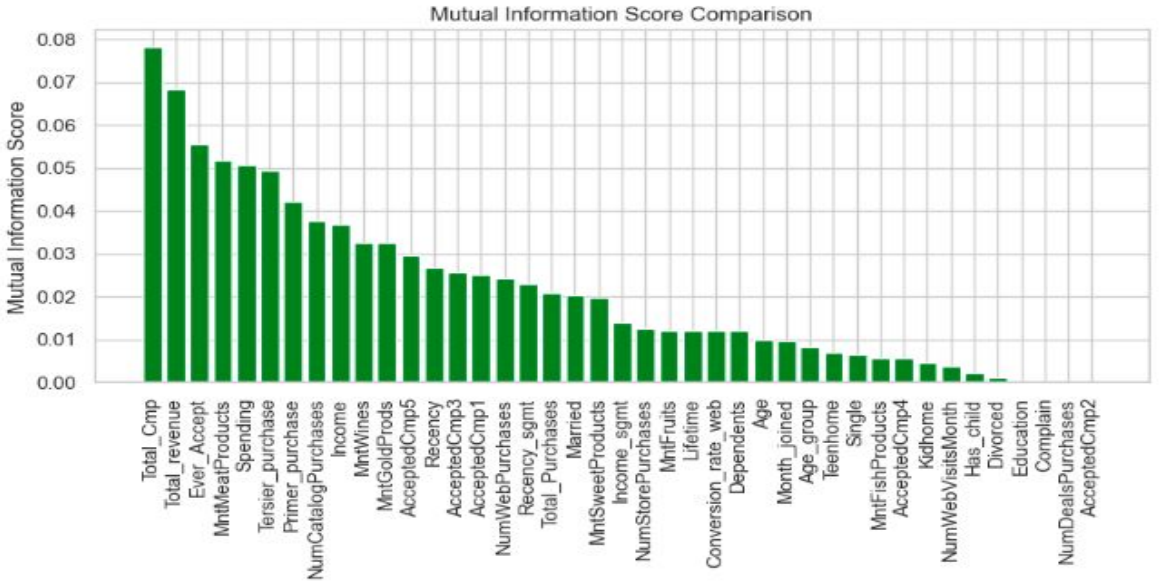
# 2. Univariate Selection

Rakamin Academy

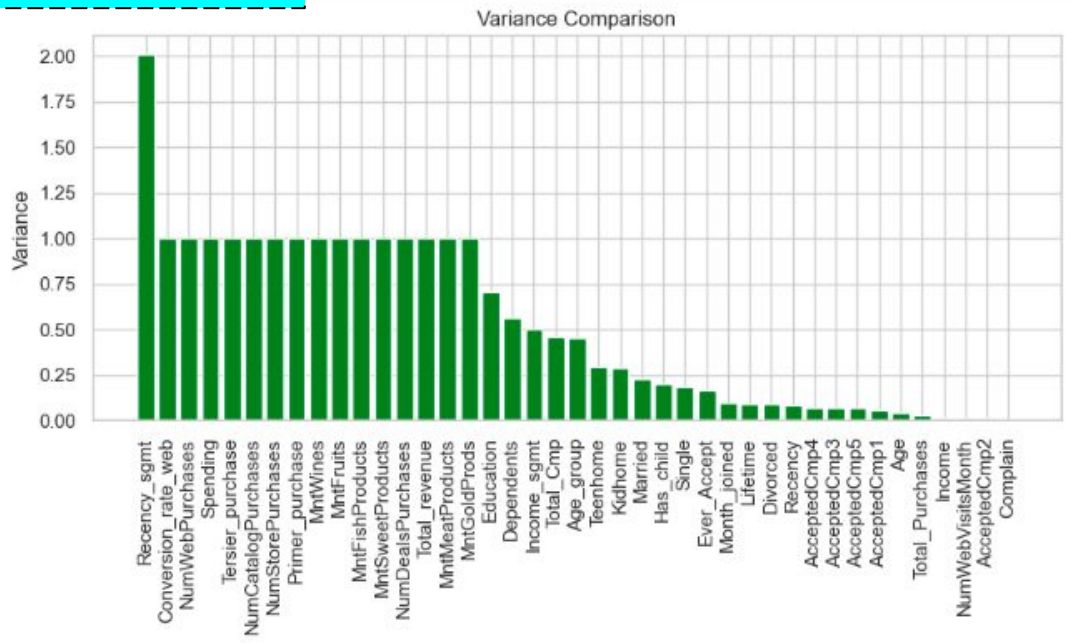
## ANOVA F-value



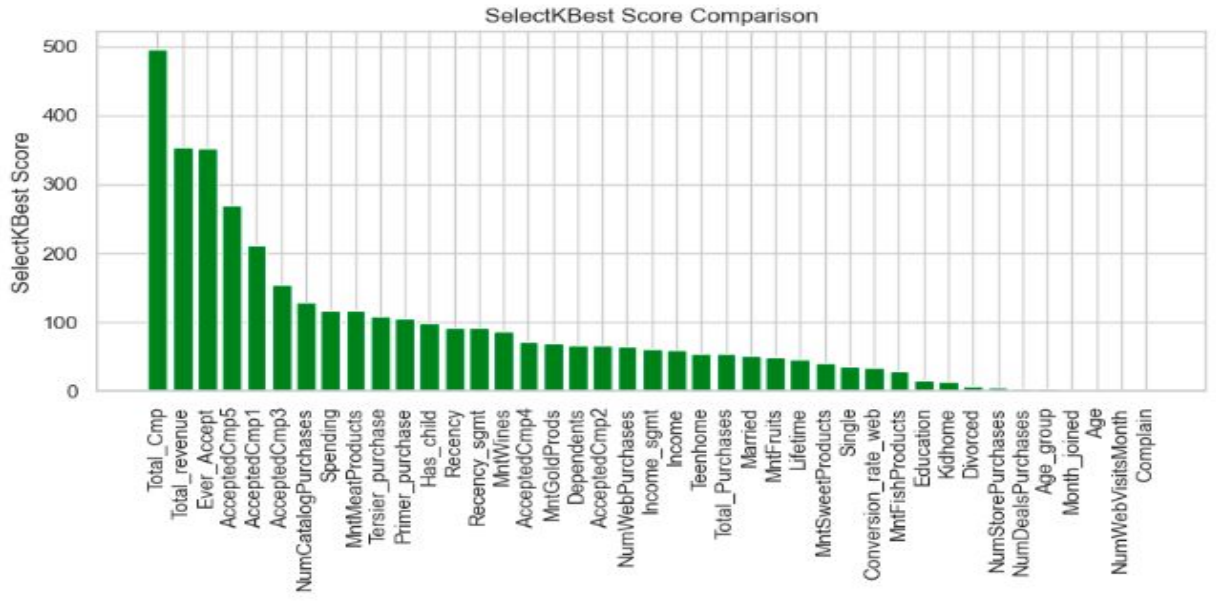

## Mutual information




## Variance Threshold




## Scikit-learn's SelectKBest



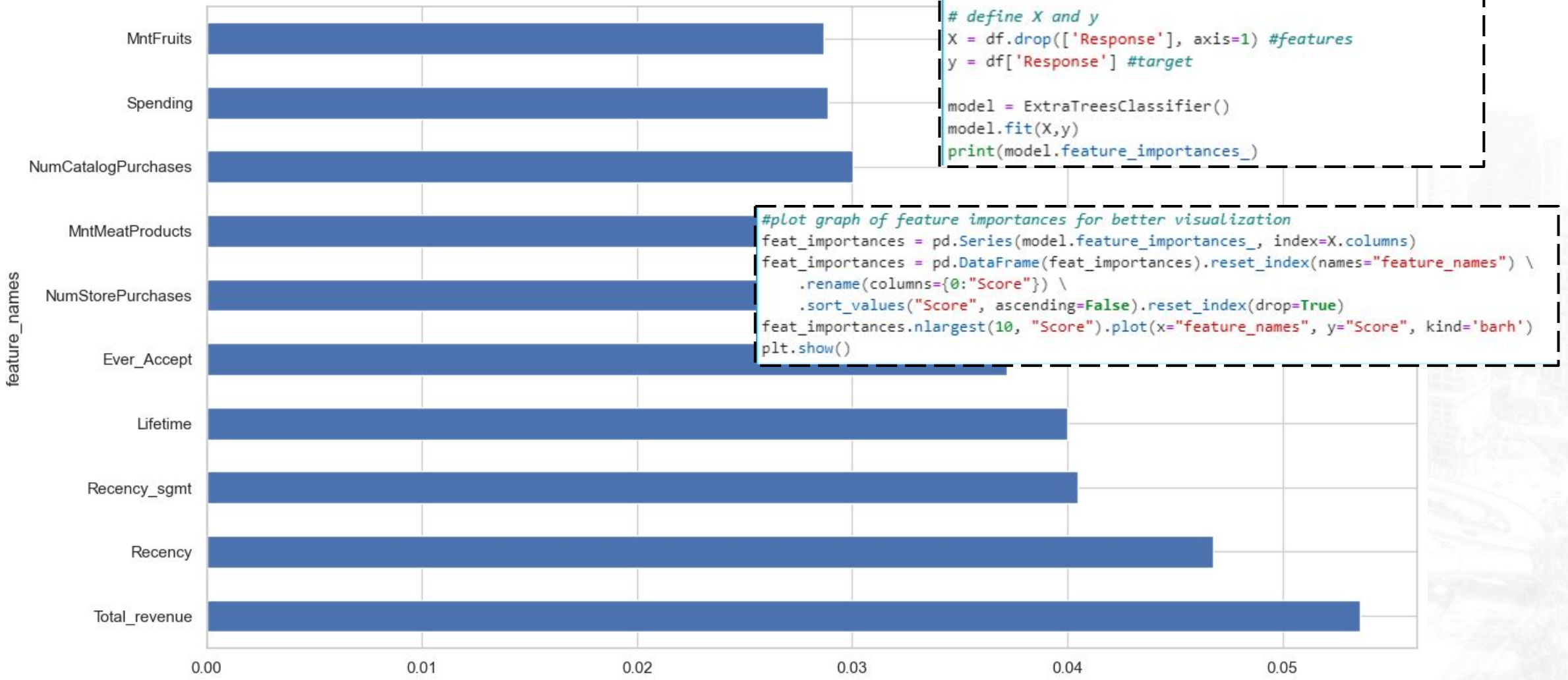
# Feature Selection

Rakamin Academy

## 3. Feature Importance

```python
from sklearn.ensemble import ExtraTreesClassifier

# define X and y
X = df.drop(['Response'], axis=1) #features
y = df['Response'] #target

model = ExtraTreesClassifier()
model.fit(X,y)
print(model.feature_importances_)
```

```python
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances = pd.DataFrame(feat_importances).reset_index(names="feature_names") \
    .rename(columns={0:"Score"}) \
    .sort_values("Score", ascending=False).reset_index(drop=True)
feat_importances.nlargest(10, "Score").plot(x="feature_names", y="Score", kind='barh')
plt.show()
```



# 4. Correlation Matrix with Heatmap

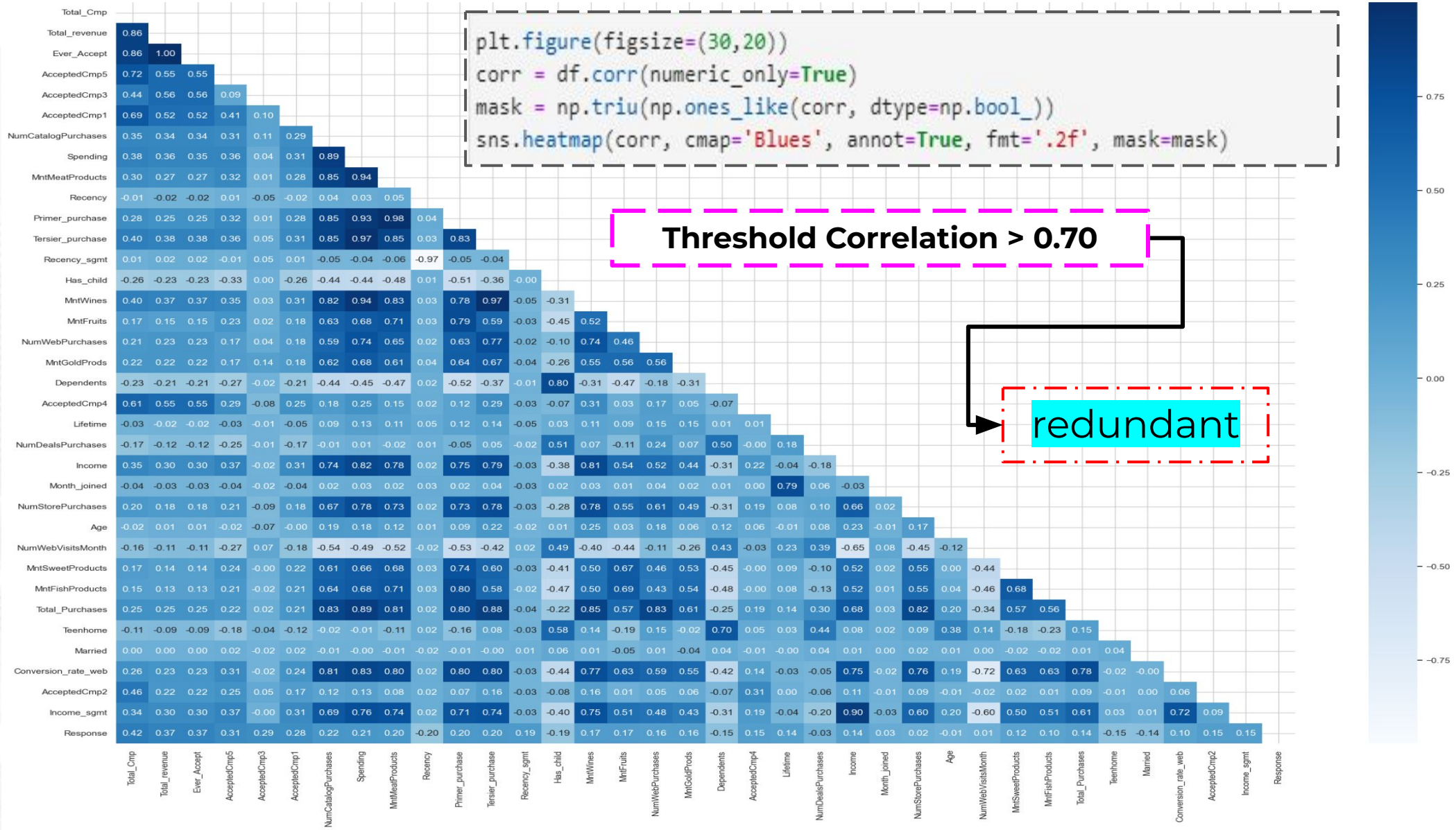
# Feature Selection

Rakamin Academy

```
plt.figure(figsize=(30,20))
corr = df.corr(numeric_only=True)
mask = np.triu(np.ones_like(corr, dtype=np.bool_))
sns.heatmap(corr, cmap='Blues', annot=True, fmt='.2f', mask=mask)
```

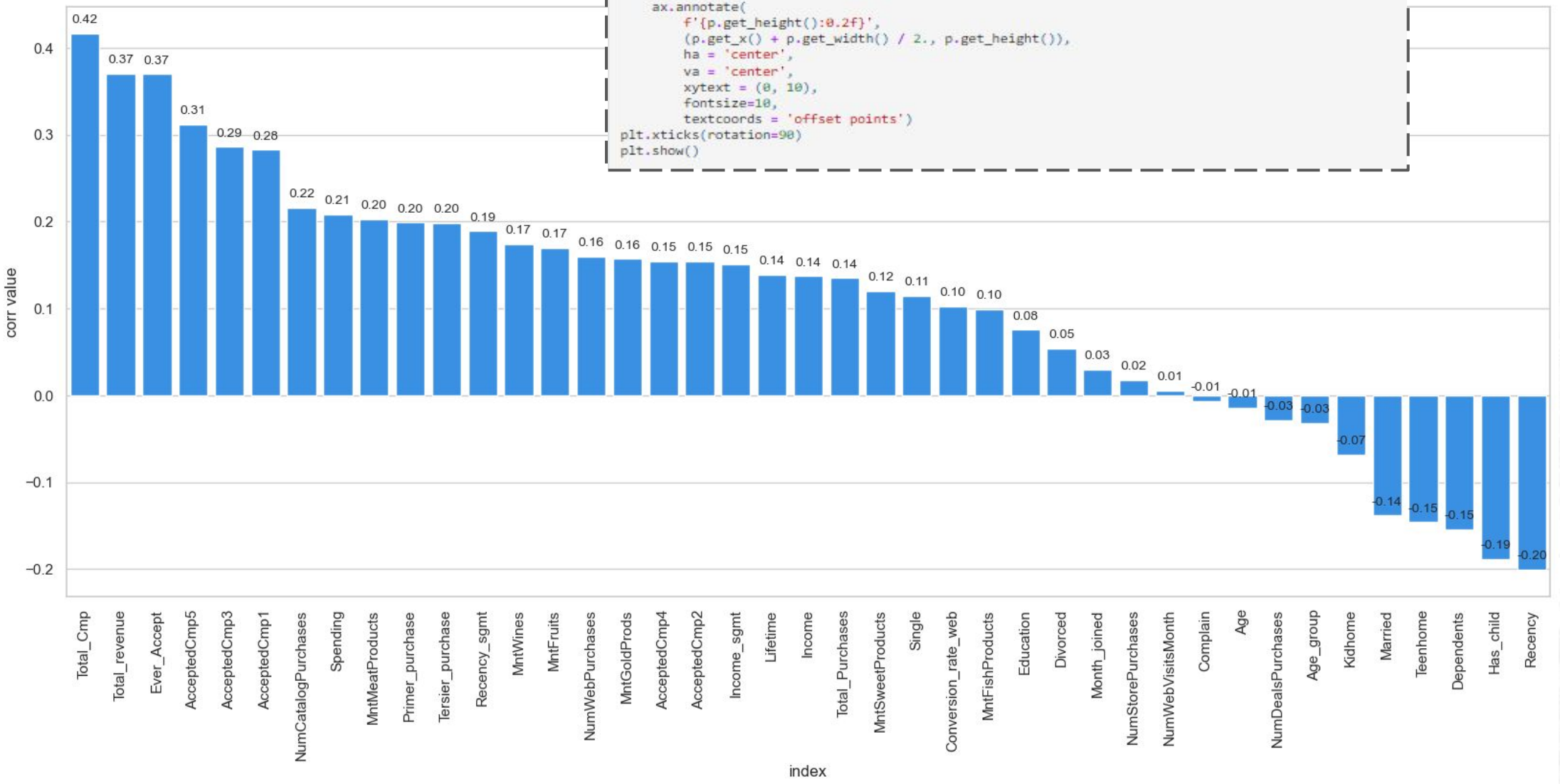
**Threshold Correlation > 0.70**

redundant



# Checking Correlation with Target (Response)

```
corr = df.corrwith(df["Response"], numeric_only=True)
corr = corr.reset_index(name='corr value')
corr = corr.sort_values('corr value', ascending=False)[1:]

plt.figure(figsize=(20, 8))
ax = sns.barplot(x='index', y="corr value", data=corr, order=corr["index"], color='dodgerblue')
for p in ax.patches:
    ax.annotate(
        f'{p.get_height():0.2f}',
        (p.get_x() + p.get_width() / 2., p.get_height()),
        ha = 'center',
        va = 'center',
        xytext = (0, 10),
        fontsize=10,
        textcoords = 'offset points')
plt.xticks(rotation=90)
plt.show()
```



## Mengambil 20 Top Features

```
for i in corr["index"].values[:20]:
    if i not in feature_importance:
        feature_importance.append(i)
feature_importance
```

**output:**

```
['Total_Cmp', 'Total_revenue',
 'Ever_Accept', 'AcceptedCmp5',
 'AcceptedCmp3', 'AcceptedCmp1',
 'NumCatalogPurchases', 'Spending',
 'MntMeatProducts', 'Recency',
 'Primer_purchase', 'Tersier_purchase',
 'Recency_sgmt', 'Has_child',
 'MntWines', 'MntFruits',
 'NumWebPurchases', 'MntGoldProds',
 'Dependents', 'AcceptedCmp4',
 'Lifetime', 'NumDealsPurchases',
 'Income', 'Month_joined',
 'NumStorePurchases', 'Age',
 'NumWebVisitsMonth', 'MntSweetProducts',
 'MntFishProducts', 'Total_Purchases',
 'Teenhome', 'Married', 'Conversion_rate_web',
'AcceptedCmp2', 'Income_sgmt']
```

# 5. Check Data Redundancy

Rakamin Academy

**Manampilkan Korelasi Feature > Threshold 0.70**

```
def corrtarget(x):
    target = "Response"
    return df[x].corr(df[target])

def corrresp(x):
    target = "Response"
    col1 = x["A"]
    col2 = x["B"]

    cor1 = df[col1].corr(df[target])
    cor2 = df[col2].corr(df[target])

    if cor1 < cor2:
        return col1
    else:
        return col2
    return col1
```

```
corr_matrix = df[feature_importance].corr()
target = "Response"

# Flatten correlation matrix.
flat_cm = corr_matrix.stack().reset_index()
flat_cm.columns = ['A', 'B', 'correlation']
flat_cm = flat_cm.loc[flat_cm.correlation < 1, :]
flat_cm = flat_cm.sort_values("correlation", ascending=False)
redundan = flat_cm[flat_cm["correlation"] >= 0.7].reset_index(drop=True)
redundan['A vs Target'] = redundan['A'].apply(lambda x: corrtarget(x))
redundan['B vs Target'] = redundan['B'].apply(lambda x: corrtarget(x))
redundan = redundan.drop_duplicates(subset=["correlation"])
redundan["drop"] = redundan.apply(corrresp, axis=1)
redundan
```

# 5. Check Data Redundancy (.....continue)

# Feature Selection

Rakamin Academy

| | A | B | correlation | A vs Target | B vs Target | drop |
|---|---|---|---|---|---|---|
| 0 | Ever_Accept | Total_revenue | 0.999982 | 0.368726 | 0.369906 | Ever_Accept |
| 2 | MntMeatProducts | Primer_purchase | 0.977667 | 0.223704 | 0.212989 | Primer_purchase |
| 4 | MntWines | Tersier_purchase | 0.976146 | 0.193849 | 0.215490 | MntWines |
| 6 | Tersier_purchase | Spending | 0.970442 | 0.215490 | 0.223961 | Tersier_purchase |
| 8 | MntMeatProducts | Spending | 0.937875 | 0.223704 | 0.223961 | MntMeatProducts |
| 10 | MntWines | Spending | 0.937196 | 0.193849 | 0.223961 | MntWines |
| 12 | Spending | Primer_purchase | 0.929771 | 0.223961 | 0.212989 | Primer_purchase |
| 14 | Spending | NumCatalogPurchases | 0.894305 | 0.223961 | 0.233715 | Spending |
| 16 | Income | Income_sgmt | 0.892218 | 0.161643 | 0.163239 | Income |
| 18 | Spending | Total_Purchases | 0.873134 | 0.223961 | 0.154715 | Total_Purchases |
| 20 | Total_revenue | Total_Cmp | 0.862955 | 0.369906 | 0.426206 | Total_revenue |
| 22 | Tersier_purchase | Total_Purchases | 0.861373 | 0.215490 | 0.154715 | Total_Purchases |
| 24 | Ever_Accept | Total_Cmp | 0.860109 | 0.368726 | 0.426206 | Ever_Accept |
| 26 | Tersier_purchase | NumCatalogPurchases | 0.856828 | 0.215490 | 0.233715 | Tersier_purchase |
| 28 | Tersier_purchase | MntMeatProducts | 0.852559 | 0.215490 | 0.223704 | Tersier_purchase |
| 30 | NumCatalogPurchases | MntMeatProducts | 0.845912 | 0.233715 | 0.223704 | MntMeatProducts |
| 110 | Income_sgmt | Tersier_purchase | 0.738756 | 0.163239 | 0.215490 | Income_sgmt |
| 112 | NumWebPurchases | MntWines | 0.737381 | 0.168142 | 0.193849 | NumWebPurchases |
| 114 | NumStorePurchases | Primer_purchase | 0.736144 | 0.049484 | 0.212989 | NumStorePurchases |
| 116 | Conversion_rate_web | Income_sgmt | 0.728599 | 0.124409 | 0.163239 | Conversion_rate_web |
| 118 | Total_Cmp | AcceptedCmp5 | 0.716298 | 0.426206 | 0.328148 | AcceptedCmp5 |
| 120 | Primer_purchase | Income_sgmt | 0.714907 | 0.212989 | 0.163239 | Income_sgmt |
| 122 | MntMeatProducts | MntFishProducts | 0.708560 | 0.223704 | 0.113616 | MntFishProducts |
| 124 | MntFruits | MntMeatProducts | 0.702425 | 0.148318 | 0.223704 | MntFruits |

**Drop Redundancy & List of feature Importances**

```
for i in list(redundan["drop"].unique()):
    feature_importance.remove(i)

feature_importance = sorted(feature_importance)
feature_importance
```

**output:**

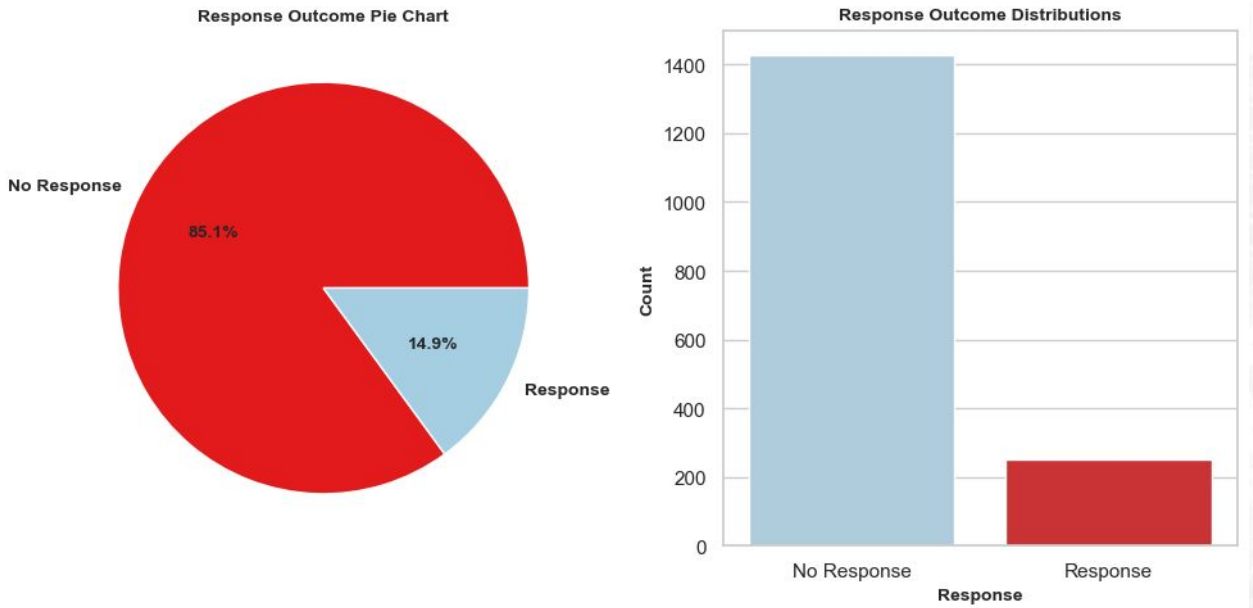
```
['AcceptedCmp1',
 'AcceptedCmp2',
 'AcceptedCmp3',
 'AcceptedCmp4',
 'Age',
 'Income',
 'Lifetime',
 'Married',
 'MntGoldProds',
 'NumCatalogPurchases',
 'NumDealsPurchases',
 'NumWebVisitsMonth',
 'Recency',
 'Recency_sgmt',
 'Teenhome',
 'Total_Cmp']
```

# Handling Imbalanced Data

Status risiko → **highly imbalanced** (15% Response & 85% No Response)

resampling

```
fig = plt.figure(figsize = (10, 5))

plt.subplot(121)
plt.pie(y_train.value_counts(),
        labels = ['No Response', 'Response'],
        autopct = '%.1f%%',
        radius = 1,
        colors=["#e31a1c", "#a6cee3"],
        textprops={'fontsize': 10, 'fontweight': 'bold'})
plt.title('Response Outcome Pie Chart', fontsize = 10, fontweight = 'bold')

plt.subplot(122)
resp = y_train.apply(lambda x: "No Response" if x == 0 else "Response")
t = sns.countplot(x=resp, palette=["#a6cee3", "#e31a1c"])
t.set_xlabel('Response', fontweight = 'bold', fontsize = 10)
t.set_ylabel('Count', fontweight = 'bold', fontsize = 10)

plt.title('Response Outcome Distributions', fontsize = 10, fontweight = 'bold')
plt.tight_layout()
```



# Handling Imbalanced Data

Rakamin Academy

## SMOTE

Status risiko highly imbalanced, dengan **15% Response dan 85% No Response**. Itu sebabnya diperlukan resampling.
**Summary :** You must apply SMOTE after splitting into training and test, not before. Doing SMOTE before is bogus and defeats the purpose of having a separate test set.

## SAMPLING

```
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler, SMOTE

print('Before OverSampling, the shape of X_train: {}'.format(X_train.shape))
print('Before OverSampling, the shape of y_train: {} \n'.format(y_train.shape))

print("Before OverSampling, counts of label '1': {}".format(sum(y_train == 1))) # Response
print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train == 0))) # No Response
```

```
# Oversampling SMOTE
sm = SMOTE(sampling_strategy=0.5, random_state = 2)
X_balanced_res, y_balanced_res = sm.fit_resample(X_train,y_train)

print('After OverSampling, the shape of X_train: {}'.format(X_balanced_res.shape))
print('After OverSampling, the shape of y_train: {} \n'.format(y_balanced_res.shape))

print("After OverSampling, counts of label '1': {}".format(sum(y_balanced_res == 1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_balanced_res == 0)))

X_train = X_balanced_res
y_train = y_balanced_res
```

**output:**

```
Before OverSampling, counts of label '1': 251
Before OverSampling, counts of label '0': 1428

After OverSampling, the shape of X_train: (2142, 16)
After OverSampling, the shape of y_train: (2142,)

After OverSampling, counts of label '1': 714
After OverSampling, counts of label '0': 1428
```

Rakamin Academy

# Stage 3

*Machine Learning Modelling & Evaluation*


# Business Flow Simulation

Rakamin Academy

## Before Model Deployment

**Lotto Mart**
Products and Type of Purchases

**All / Random Customers**
There are no Target Customers

**Marketing Campaign**

**Low Response Rate**
**Cost > Revenue (-%NPM)**

## After Model Deployment

**Lotto Mart**
Products and Type of Purchases

**Predicting Customers with Responses**

**Classification Models**

N → **No Marketing Campaigns**
Optimizing Marketing Campaign
Providing Other Treatment

Y → **Customer Segmentation RFM**

**Clustering Models**

**Customer Cluster Results**
Potential Taken

**Check Behavior**
**Business Treatment**
**Recommendations**

**Marketing Campaigns**
Recommended

**More Targeted**
**Increase Response Rate**
**Cost < Revenue (%NPM)**

# Classification Models

# Machine Learning Techniques

## Jenis-Jenis Algoritma yang digunakan

Berikut beberapa algoritma yang akan dilakukan pengujian untuk **menentukan model terbaik** dalam **memprediksi respon pelanggan pada campaign marketing berikutnya**:

1. Decision Tree
2. **Random Forest**
3. Logistic Regression
4. Gaussian Naive Bayes
5. K-Nearest Neighbor
6. MLP Classifier (Neural Network)
7. Adaboost Classifier
8. XGBoost Classifier
9. Gradient Boosting Classifier
10. Support Vector Machine

## Parameter Evaluasi Model

Parameter evaluasi model yang akan digunakan berfokus pada ketiga hal berikut:

- **Precision,** digunakan sebagai parameter pengukuran utama
  - Meningkatkan Response Rate
  - Mereduksi False Positif (Customer yang diprediksi akan merespon, namun kenyataannya tidak)
- **Recall,** digunakan sebagai parameter pengukuran sekunder
  - Mengoptimalkan Revenue Rate
  - Mereduksi False Negative (Customer yang diprediksi tidak mengikuti campaign, namun pada kenyataannya berkeinginan ikut campaign)
- **F1 Score,** digunakan untuk melakukan pemeriksaan nilai skor positif dan negatif (Imbalanced Data)

# Modelling

## 1. Decision Tree

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.997000 | 0.838000 | 0.159000 |
| 1 | Precision | 1.000000 | 0.456000 | 0.544000 |
| 2 | Recall | 0.990000 | 0.494000 | 0.496000 |
| 3 | F1 Score | 0.995000 | 0.474000 | 0.521000 |
| 4 | F1 Score (crossval) | 0.972000 | 0.488000 | 0.484000 |
| 5 | ROC AUC | 1.000000 | 0.699000 | 0.301000 |
| 6 | ROC AUC (crossval) | 1.000000 | 0.718000 | 0.282000 |

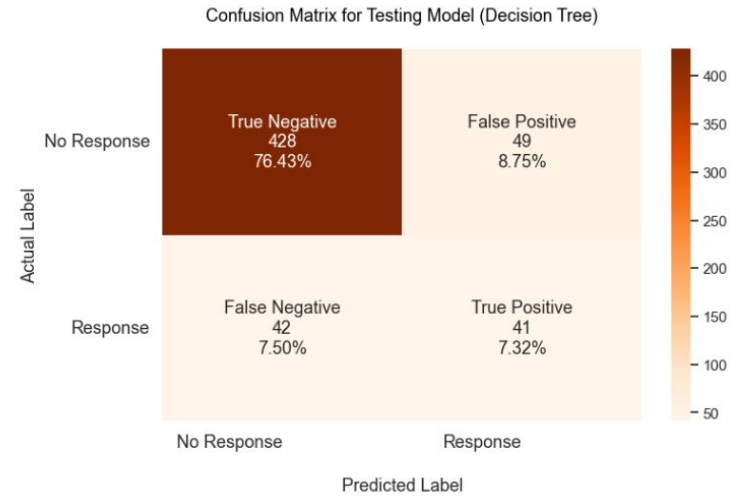
**Observation:**

- Precision dan Recall memiliki gap cukup besar >0.40
- F1 Score test dengan gap cukup besar > 0.50

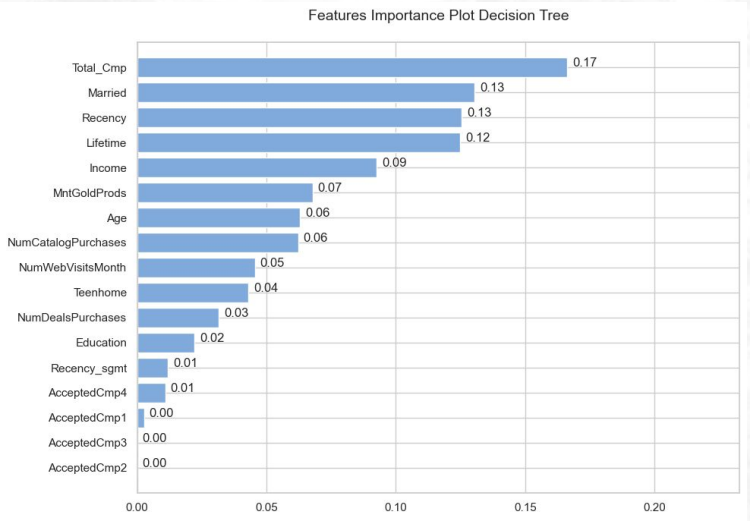
```
Training Accuracy: 99.67 %
Testing Accuracy: 83.75 %
```

## Performance of Testing Model

```
Classification Report Testing Model (Decision Tree):

Accuracy = 0.838
Precision = 0.456
Recall = 0.494
F1 Score = 0.474
Cross Val F1 (k=5) = 0.488
ROC AUC = 0.699
Cross Val ROC AUC (k=5) = 0.718

              precision    recall  f1-score   support

           0       0.91      0.90      0.90       477
           1       0.46      0.49      0.47        83

    accuracy                           0.84       560
   macro avg       0.68      0.70      0.69       560
weighted avg       0.84      0.84      0.84       560

==== Actual Data (Test) =====
Total = 560
No Response = 477
Response = 83
==== Predicted Data (Test) =====
TP = 41, FP = 49, TN = 428, FN = 42
Predictly Correct = 469
Predictly Wrong = 91
```



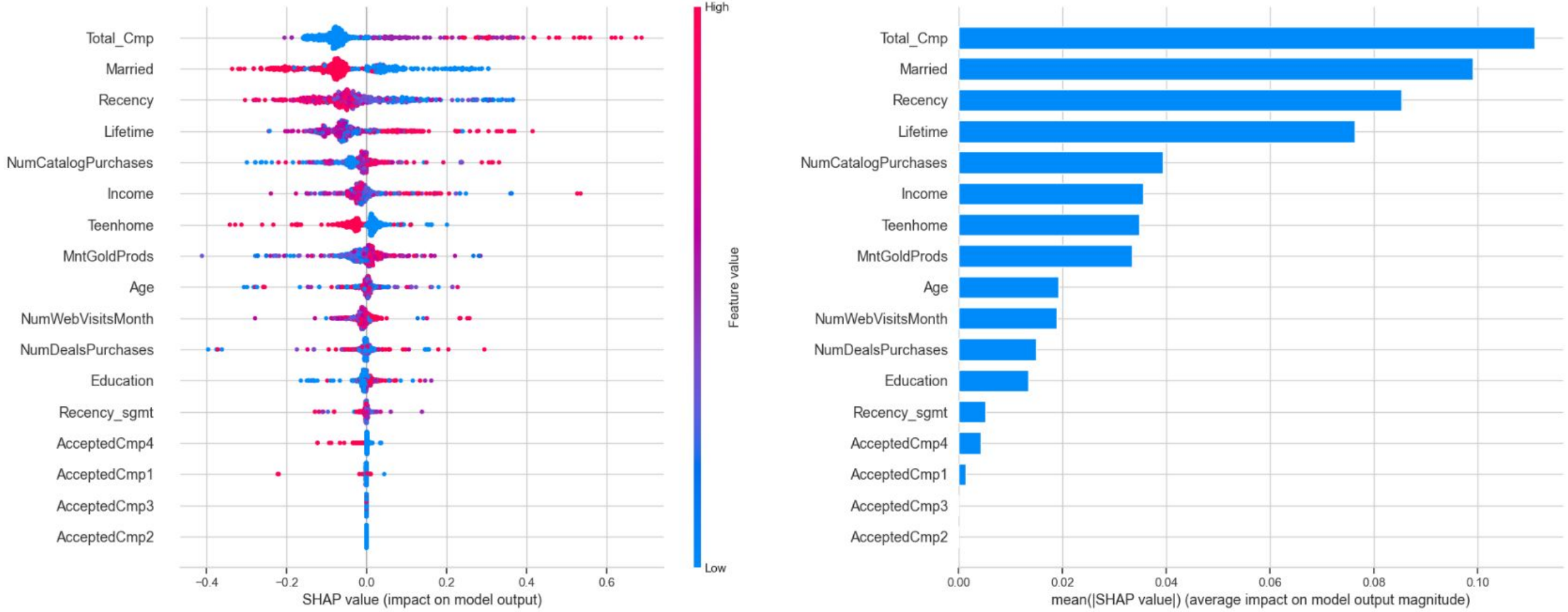




# Modelling

## 1. Decision Tree

## Performance of Testing Model



# Modelling

## 2. Random Forest

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.997000 | 0.909000 | 0.088000 |
| 1 | Precision | 0.996000 | 0.786000 | 0.210000 |
| 2 | Recall | 0.994000 | 0.530000 | 0.464000 |
| 3 | F1 Score | 0.995000 | 0.633000 | 0.362000 |
| 4 | F1 Score (crossval) | 0.972000 | 0.519000 | 0.453000 |
| 5 | ROC AUC | 1.000000 | 0.905000 | 0.095000 |
| 6 | ROC AUC (crossval) | 1.000000 | 0.896000 | 0.104000 |

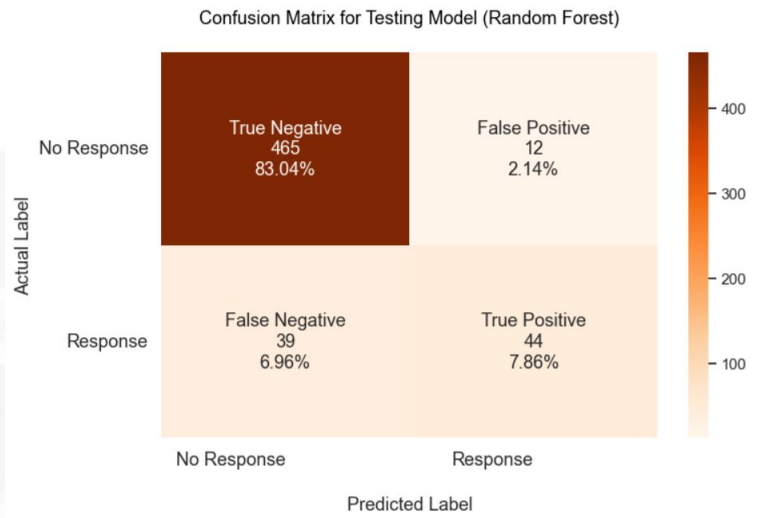
**Observation:**

- Precision memiliki gap lumayan kecil 0.21
- Recall memiliki gap yang agak besar 0.46
- F1 Score memiliki gap lumayan agak kecil 0.36

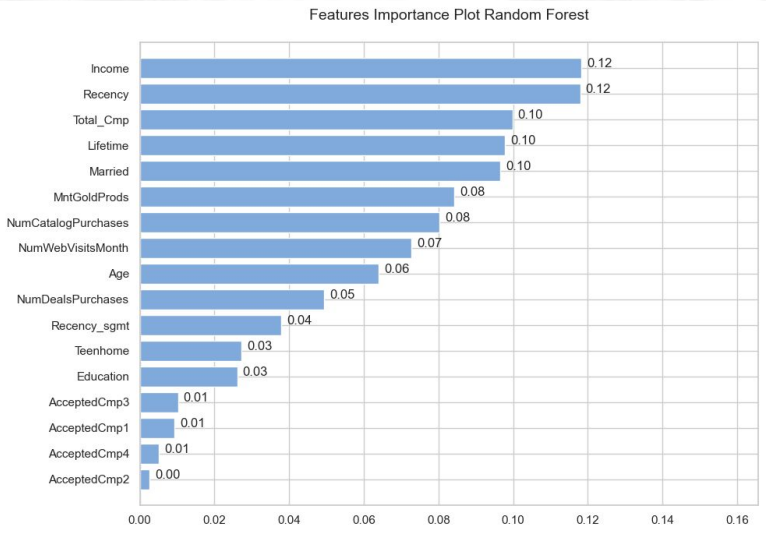
```
Training Accuracy: 99.67 %
Test Accuracy: 90.89 %
```

## Performance of Testing Model

```
Classification Report Testing Model (Random Forest):

Accuracy = 0.909
Precision = 0.786
Recall = 0.53
F1 Score = 0.633
Cross Val F1 (k=5) = 0.519
ROC AUC = 0.905
Cross Val ROC AUC (k=5) = 0.896

              precision    recall  f1-score   support

           0       0.92      0.97      0.95       477
           1       0.79      0.53      0.63        83

    accuracy                           0.91       560
   macro avg       0.85      0.75      0.79       560
weighted avg       0.90      0.91      0.90       560

==== Actual Data (Test) =====
Total = 560
No Response = 477
Response = 83
==== Predicted Data (Test) =====
TP = 44, FP = 12, TN = 465, FN = 39
Predictly Correct = 509
Predictly Wrong = 51
```



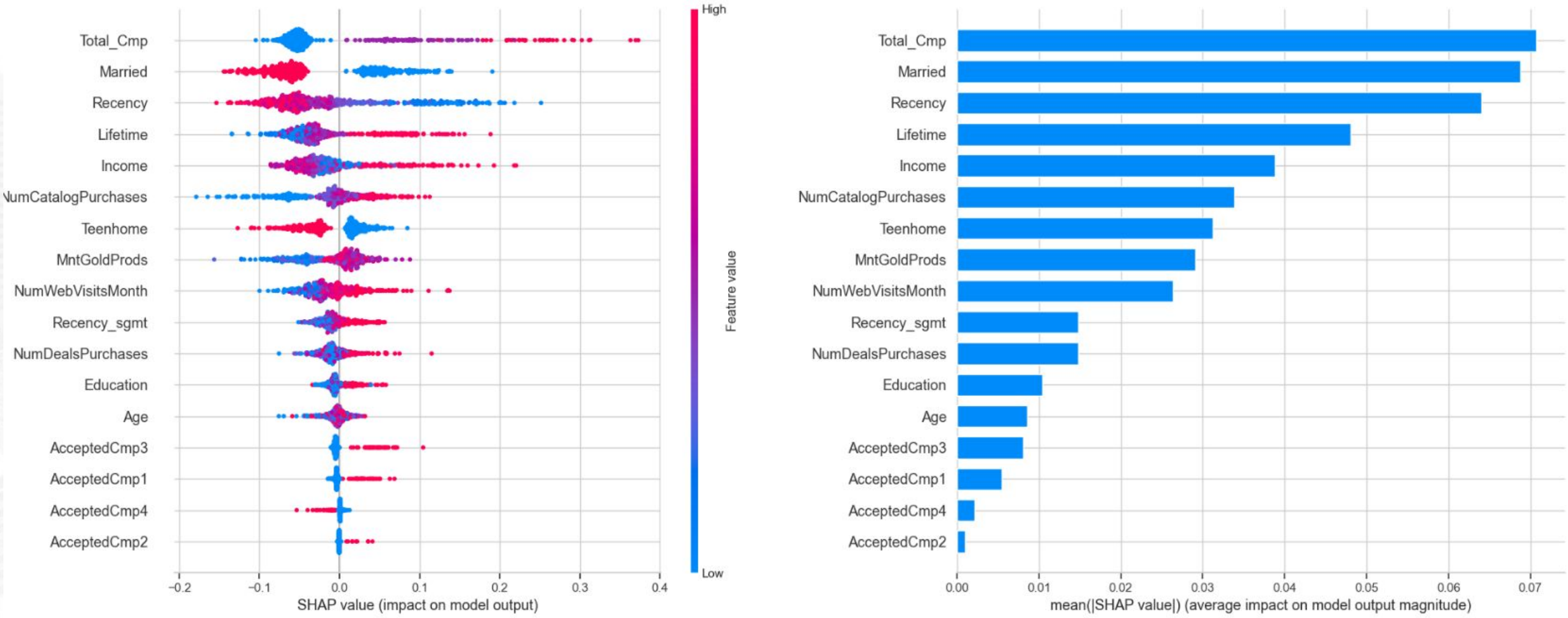




# Modelling

## 2. Random Forest

### Performance of Testing Model



# Modelling

## 3. Logistic Regression

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.813000 | 0.861000 | -0.048000 |
| 1 | Precision | 0.750000 | 0.523000 | 0.227000 |
| 2 | Recall | 0.660000 | 0.675000 | -0.015000 |
| 3 | F1 Score | 0.702000 | 0.589000 | 0.113000 |
| 4 | F1 Score (crossval) | 0.536000 | 0.510000 | 0.026000 |
| 5 | ROC AUC | 0.887000 | 0.900000 | -0.013000 |
| 6 | ROC AUC (crossval) | 0.891000 | 0.885000 | 0.006000 |

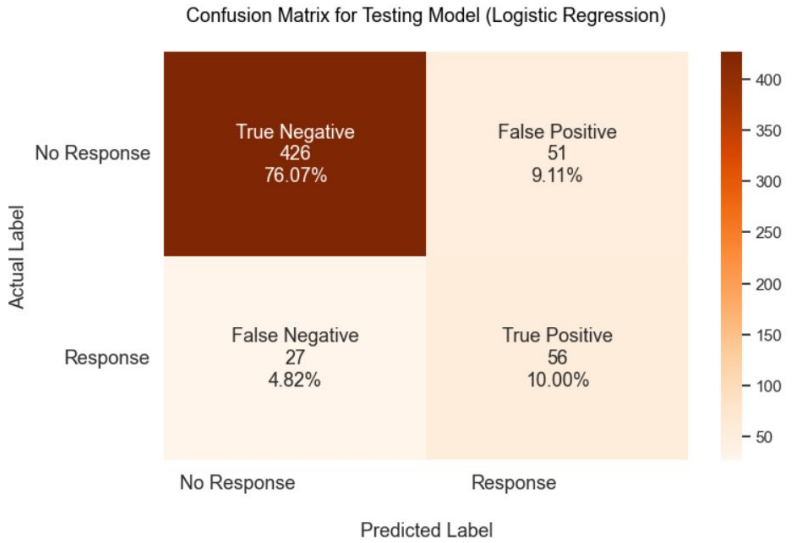
**Observation:**

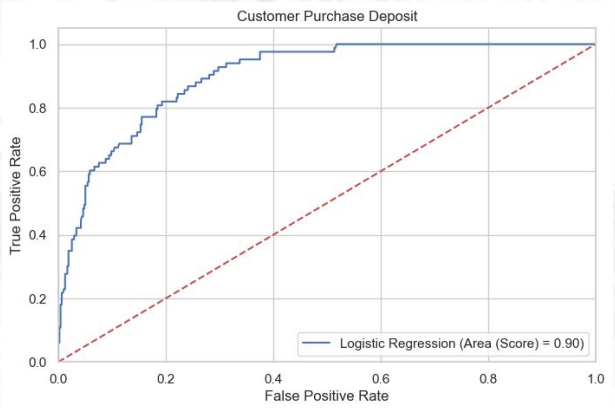
- Precision memiliki gap yang luamyan kecil 0.22
- Recall memiliki gap cukup kecil under -0.01
- F1 Score memiliki gap cukup kecil 0.11

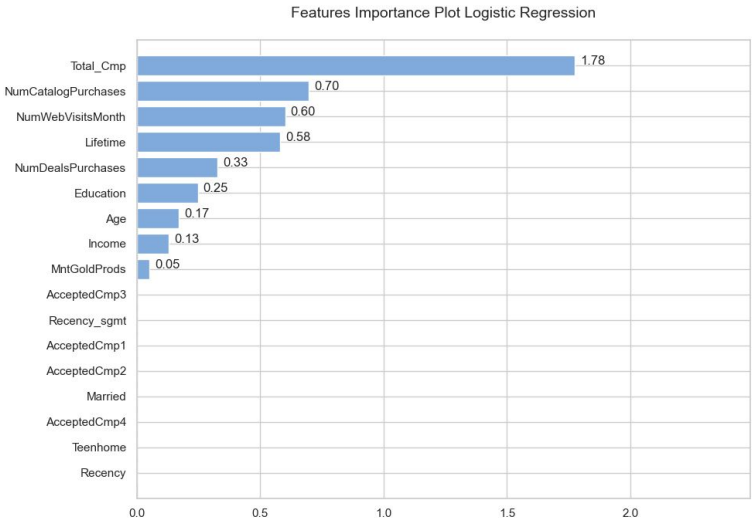
```
Training Accuracy: 81.33 %
Test Accuracy: 86.07 %
```

## Performance of Testing Model

```
Classification Report Testing Model (Logistic Regression):

Accuracy = 0.861
Precision = 0.523
Recall = 0.675
F1 Score = 0.589
Cross Val F1 (k=5) = 0.51
ROC AUC = 0.9
Cross Val ROC AUC (k=5) = 0.885

              precision    recall  f1-score   support

           0       0.94      0.89      0.92       477
           1       0.52      0.67      0.59        83

    accuracy                           0.86       560
   macro avg       0.73      0.78      0.75       560
weighted avg       0.88      0.86      0.87       560

==== Actual Data (Test) =====
Total = 560
No Response = 477
Response = 83
==== Predicted Data (Test) =====
TP = 56, FP = 51, TN = 426, FN = 27
Predictly Correct = 482
Predictly Wrong = 78
```



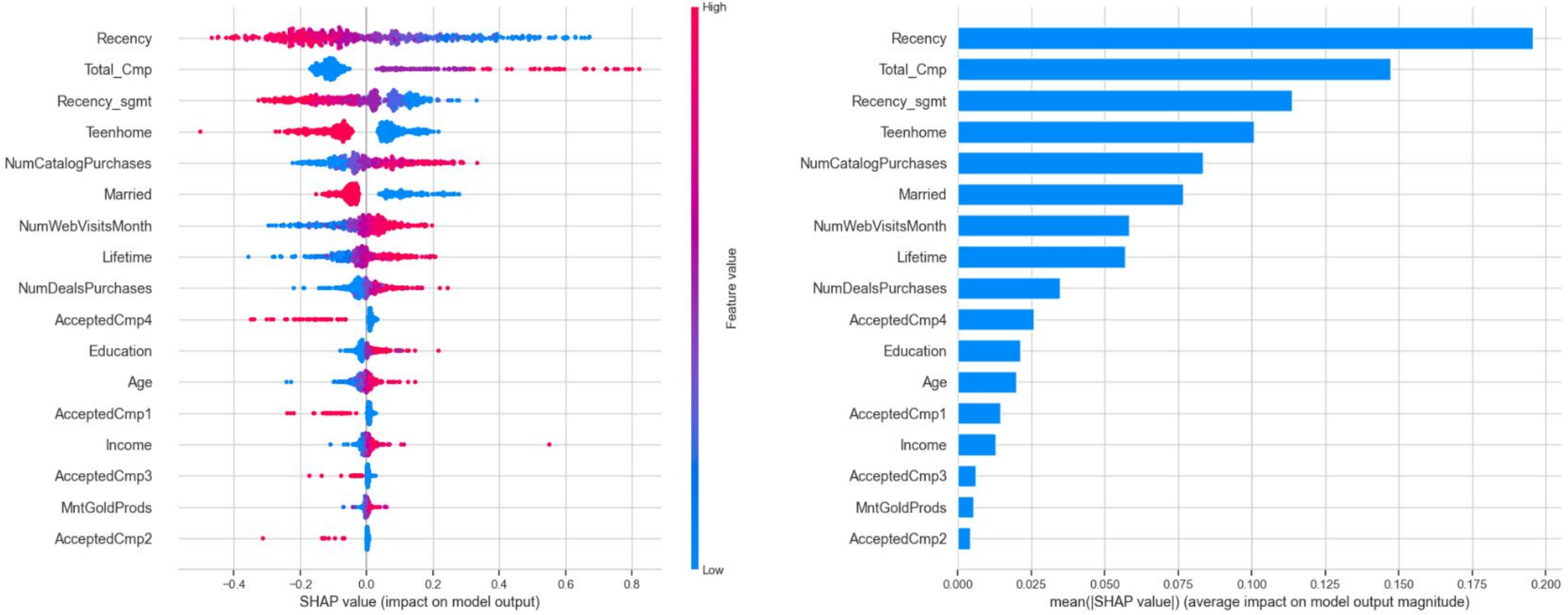



# Modelling

## 3. Logistic Regression

## Performance of Testing Model



# Modelling

## 4. Naive Bayes

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.735000 | 0.830000 | -0.095000 |
| 1 | Precision | 0.658000 | 0.441000 | 0.217000 |
| 2 | Recall | 0.429000 | 0.542000 | -0.113000 |
| 3 | F1 Score | 0.519000 | 0.486000 | 0.033000 |
| 4 | F1 Score (crossval) | 0.469000 | 0.451000 | 0.018000 |
| 5 | ROC AUC | 0.807000 | 0.838000 | -0.031000 |
| 6 | ROC AUC (crossval) | 0.828000 | 0.824000 | 0.004000 |

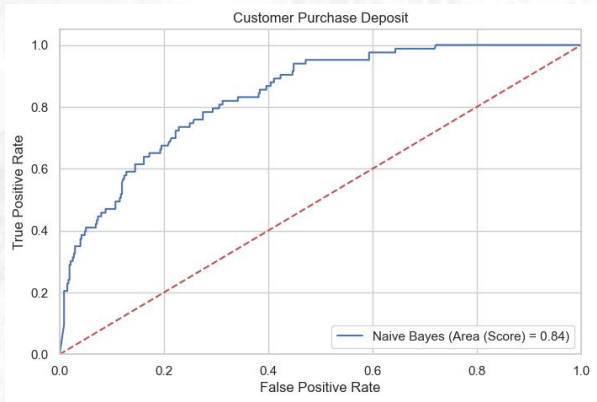
**Observation:**

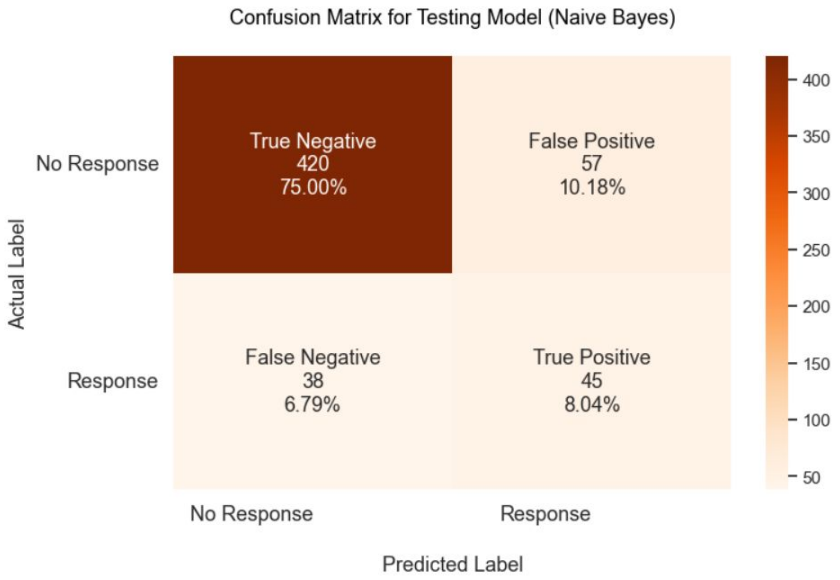
- Precision memiliki gap lumayan kecil 0.21
- Recall memiliki gap cukup kecil under -0.11
- F1 Score memiliki gap yang sangat kecil 0.03

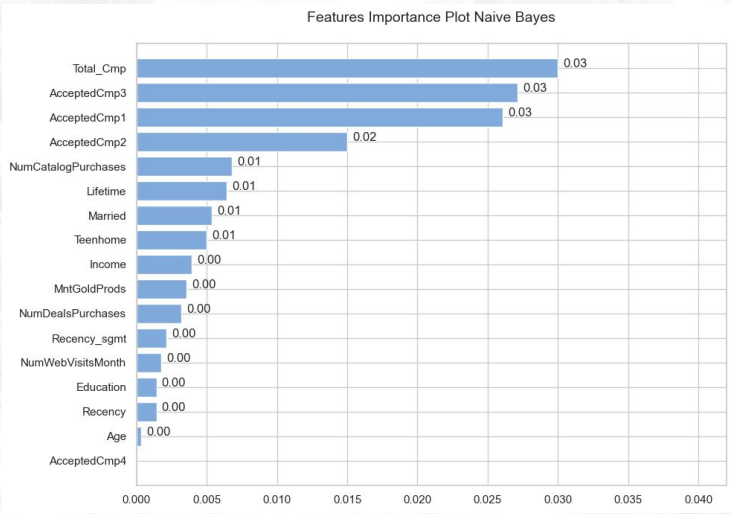
```
Training Accuracy: 73.53 %
Test Accuracy: 83.04 %
```

## Performance of Testing Model

```
Classification Report Testing Model (Naive Bayes):

Accuracy = 0.83
Precision = 0.441
Recall = 0.542
F1 Score = 0.486
Cross Val F1 (k=5) = 0.451
ROC AUC = 0.838
Cross Val ROC AUC (k=5) = 0.824

              precision    recall  f1-score   support

           0       0.92      0.88      0.90       477
           1       0.44      0.54      0.49        83

    accuracy                           0.83       560
   macro avg       0.68      0.71      0.69       560
weighted avg       0.85      0.83      0.84       560

==== Actual Data (Test) =====
Total = 560
No Response = 477
Response = 83
==== Predicted Data (Test) =====
TP = 45, FP = 57, TN = 420, FN = 38
Predictly Correct = 465
Predictly Wrong = 95
```



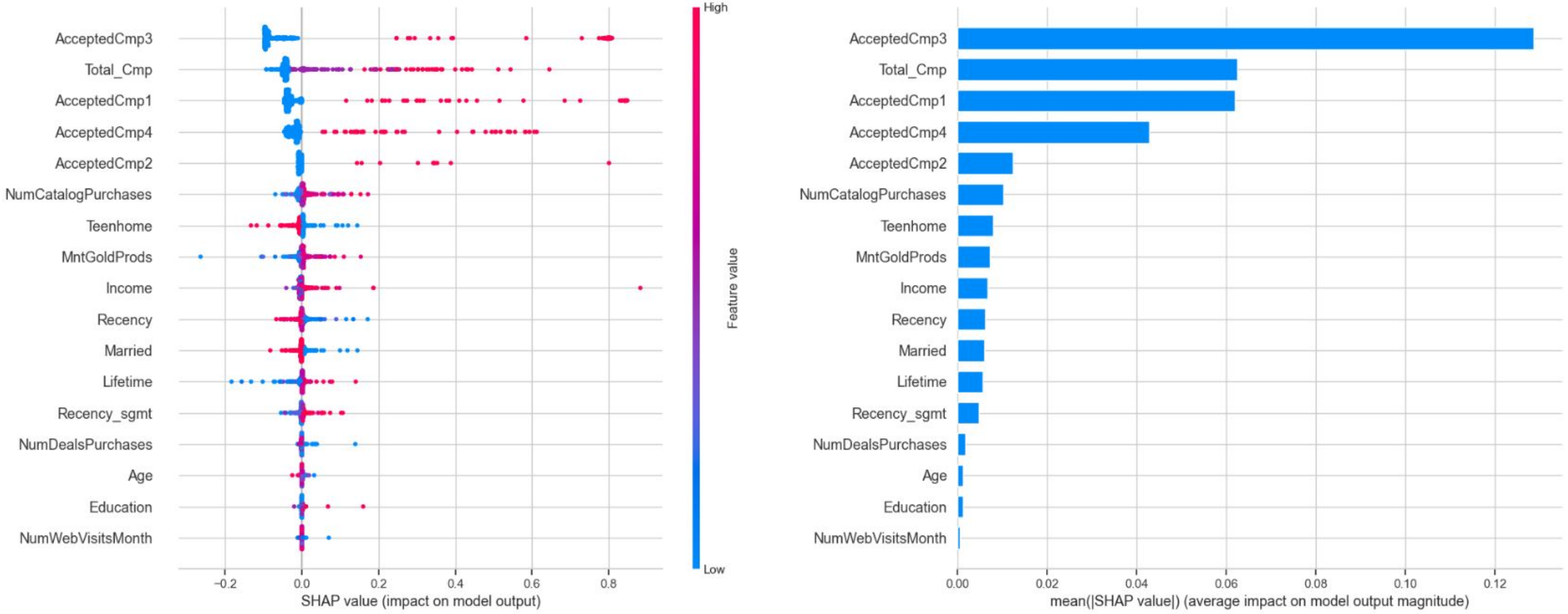



# Modelling

## 4. Naive Bayes

### Performance of Testing Model



# Modelling

## 5. K-Nearest Neigbors

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.916000 | 0.845000 | 0.071000 |
| 1 | Precision | 0.819000 | 0.482000 | 0.337000 |
| 2 | Recall | 0.959000 | 0.639000 | 0.320000 |
| 3 | F1 Score | 0.884000 | 0.549000 | 0.335000 |
| 4 | F1 Score (crossval) | 0.605000 | 0.400000 | 0.205000 |
| 5 | ROC AUC | 0.982000 | 0.842000 | 0.140000 |
| 6 | ROC AUC (crossval) | 0.949000 | 0.810000 | 0.139000 |

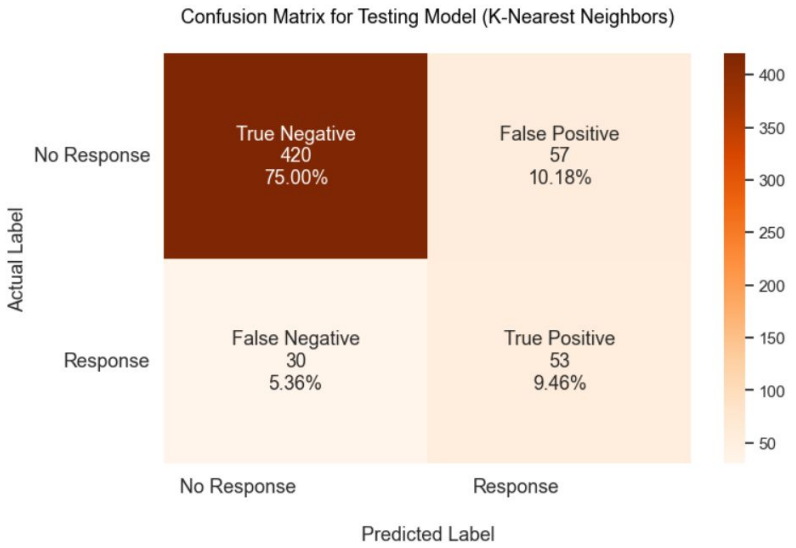
**Observation:**

- Precision memiliki gap lumayan agak besar 0.33
- Recall memiliki gap lumayan agak besar 0.32
- F1 Score memiliki gap lumayan agak besar 0.33

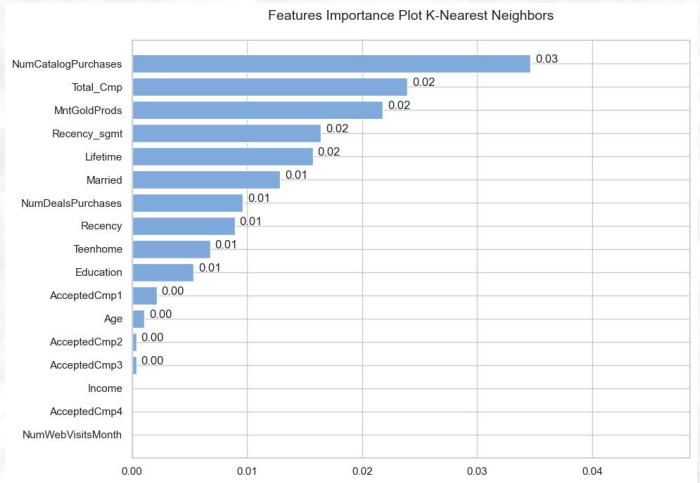
```
Training Accuracy: 91.6 %
Test Accuracy: 84.46 %
```

## Performance of Testing Model

```
Classification Report Testing Model (K-Nearest Neighbors):

Accuracy = 0.845
Precision = 0.482
Recall = 0.639
F1 Score = 0.549
Cross Val F1 (k=5) = 0.4
ROC AUC = 0.842
Cross Val ROC AUC (k=5) = 0.81

              precision    recall  f1-score   support

           0       0.93      0.88      0.91       477
           1       0.48      0.64      0.55        83

    accuracy                           0.84       560
   macro avg       0.71      0.76      0.73       560
weighted avg       0.87      0.84      0.85       560

==== Actual Data (Test) =====
Total = 560
No Response = 477
Response = 83
==== Predicted Data (Test) =====
TP = 53, FP = 57, TN = 420, FN = 30
Predictly Correct = 473
Predictly Wrong = 87
```







# Modelling

## 6. MLP Classifier (Neutral Network)

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.995000 | 0.846000 | 0.149000 |
| 1 | Precision | 0.992000 | 0.483000 | 0.509000 |
| 2 | Recall | 0.993000 | 0.518000 | 0.475000 |
| 3 | F1 Score | 0.992000 | 0.500000 | 0.492000 |
| 4 | F1 Score (crossval) | 0.874000 | 0.558000 | 0.316000 |
| 5 | ROC AUC | 1.000000 | 0.830000 | 0.170000 |
| 6 | ROC AUC (crossval) | 0.990000 | 0.887000 | 0.103000 |

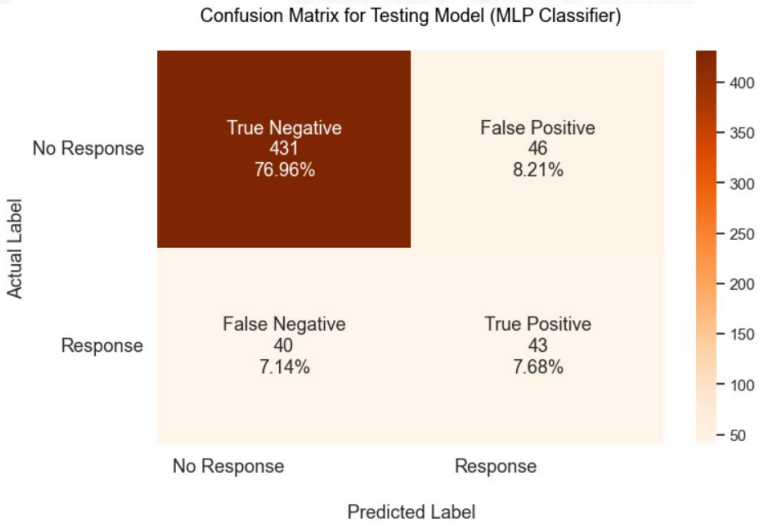
**Observation:**

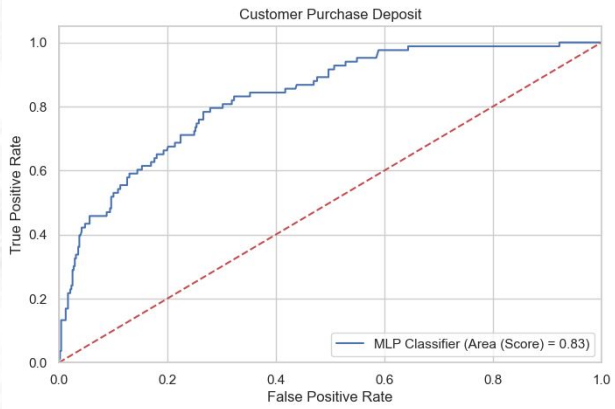
- Precision memiliki gap lumayan besar 0.50
- Recall memiliki gap cukup besar 0.47
- F1 Score memiliki gap cukup besar 0.49

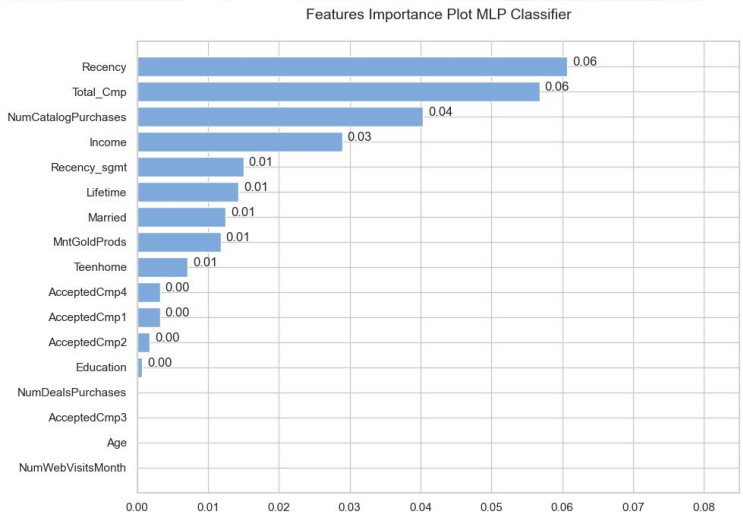
```
Training Accuracy: 99.49 %
Test Accuracy: 84.64 %
```

## Performance of Testing Model

```
Classification Report Testing Model (MLP Classifier):

Accuracy = 0.846
Precision = 0.483
Recall = 0.518
F1 Score = 0.5
Cross Val F1 (k=5) = 0.558
ROC AUC = 0.83
Cross Val ROC AUC (k=5) = 0.887

              precision    recall  f1-score   support

           0       0.92      0.90      0.91       477
           1       0.48      0.52      0.50        83

    accuracy                           0.85       560
   macro avg       0.70      0.71      0.70       560
weighted avg       0.85      0.85      0.85       560

==== Actual Data (Test) =====
Total = 560
No Response = 477
Response = 83
==== Predicted Data (Test) =====
TP = 43, FP = 46, TN = 431, FN = 40
Predictly Correct = 474
Predictly Wrong = 86
```



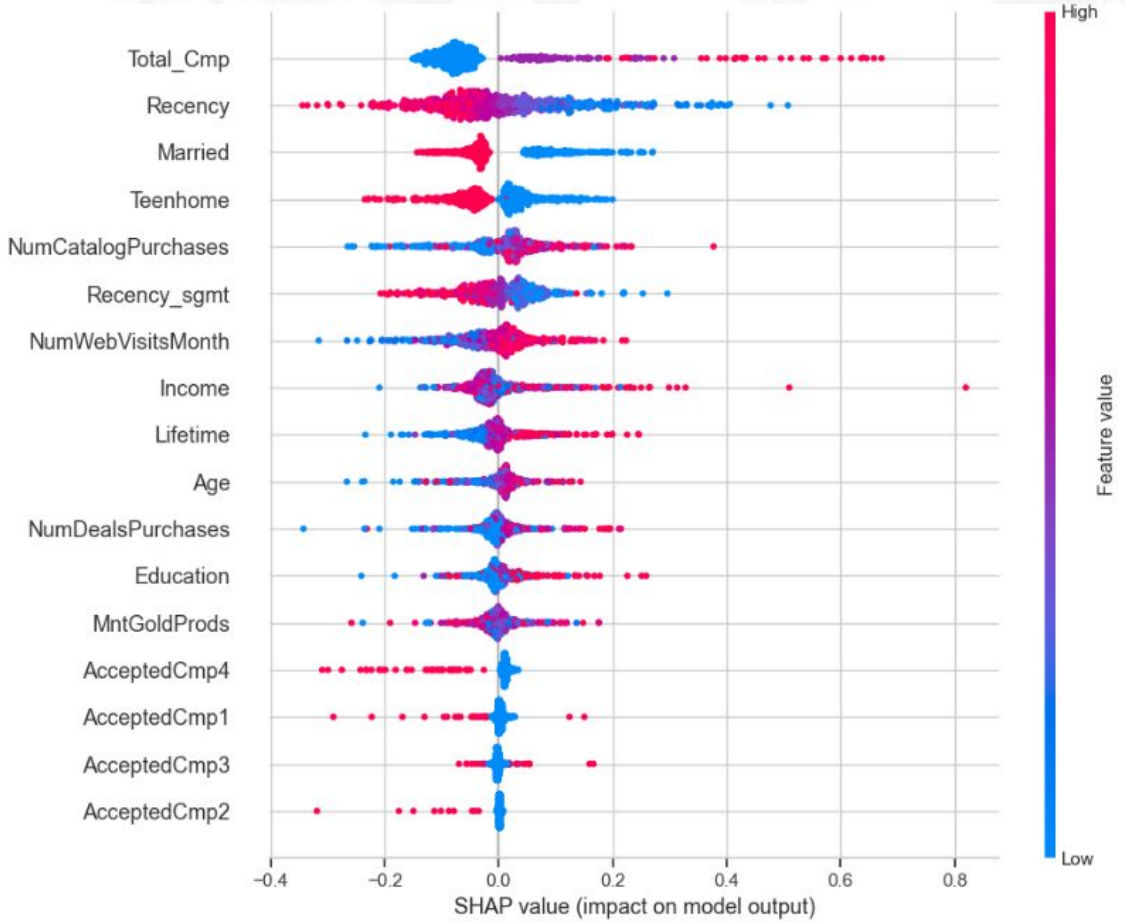



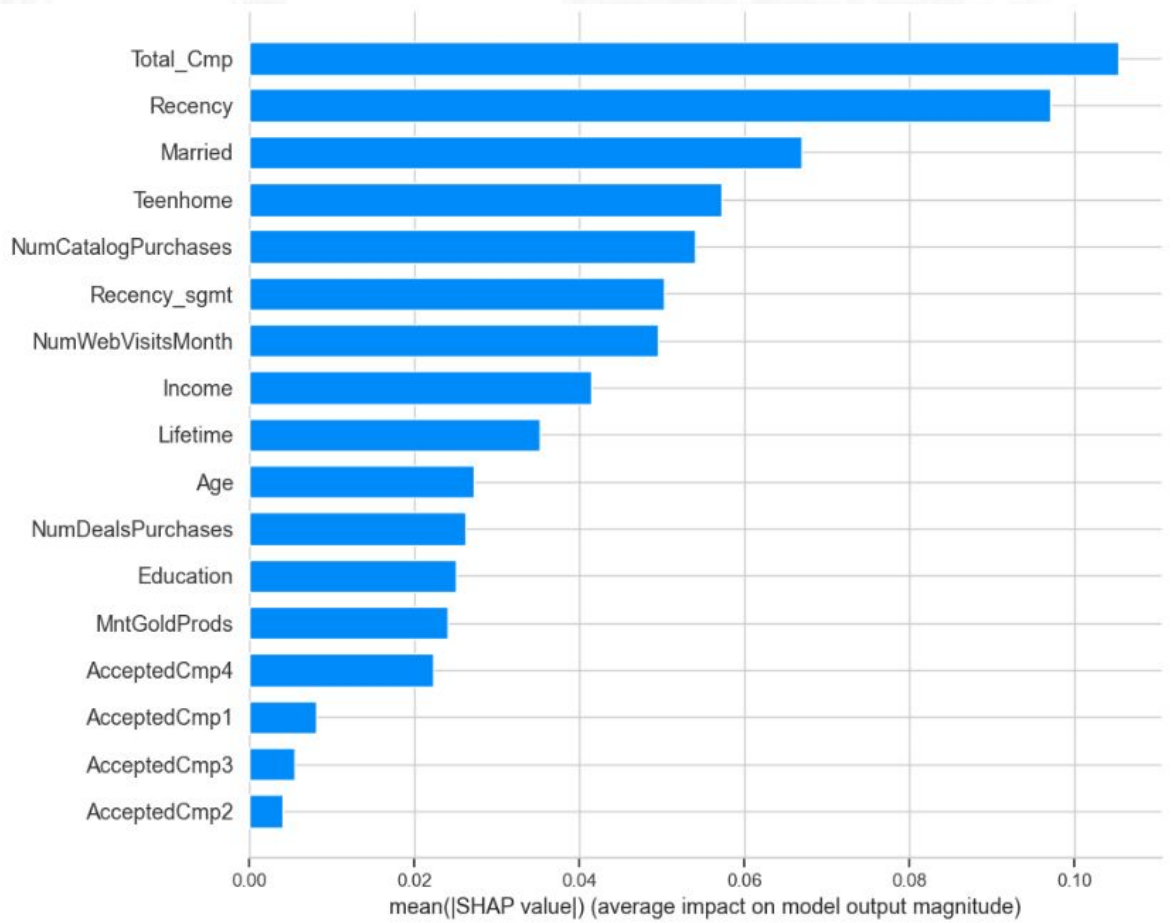



# Modelling

## 6. MLP Classifier (Neutral Network)

## Performance of Testing Model



# Modelling

## 7. Adaboost Classifier

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.873000 | 0.855000 | 0.018000 |
| 1 | Precision | 0.844000 | 0.511000 | 0.333000 |
| 2 | Recall | 0.759000 | 0.578000 | 0.181000 |
| 3 | F1 Score | 0.799000 | 0.542000 | 0.257000 |
| 4 | F1 Score (crossval) | 0.605000 | 0.539000 | 0.066000 |
| 5 | ROC AUC | 0.948000 | 0.892000 | 0.056000 |
| 6 | ROC AUC (crossval) | 0.929000 | 0.885000 | 0.044000 |

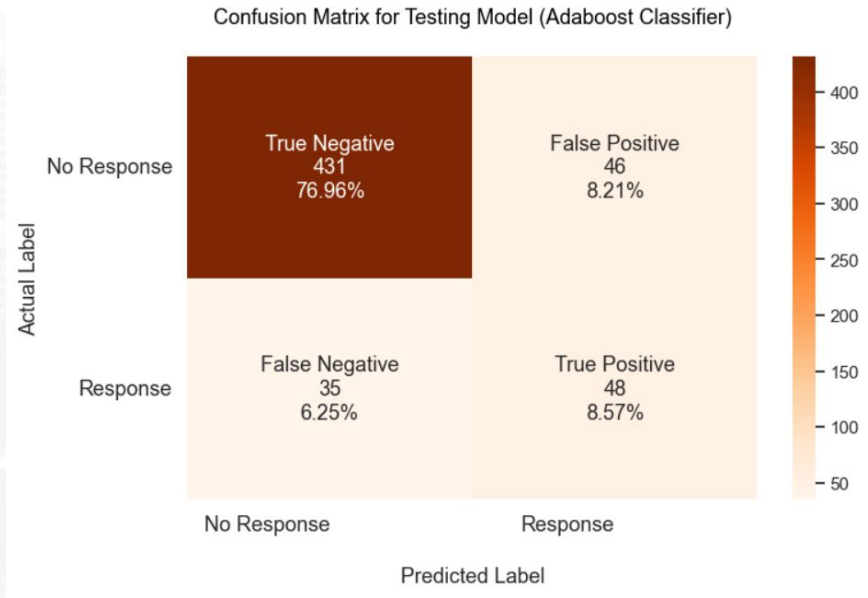
**Observation:**

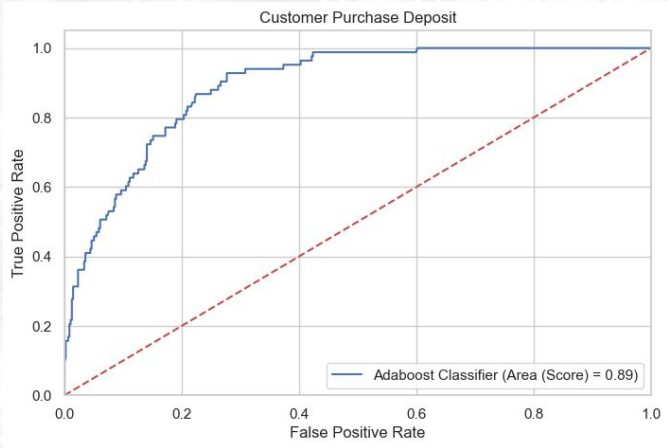
- Precision memiliki gap agak besar 0.33
- Recall memiliki gap cukup kecil 0.18
- F1 Score memiliki gap agak kecil 0.25

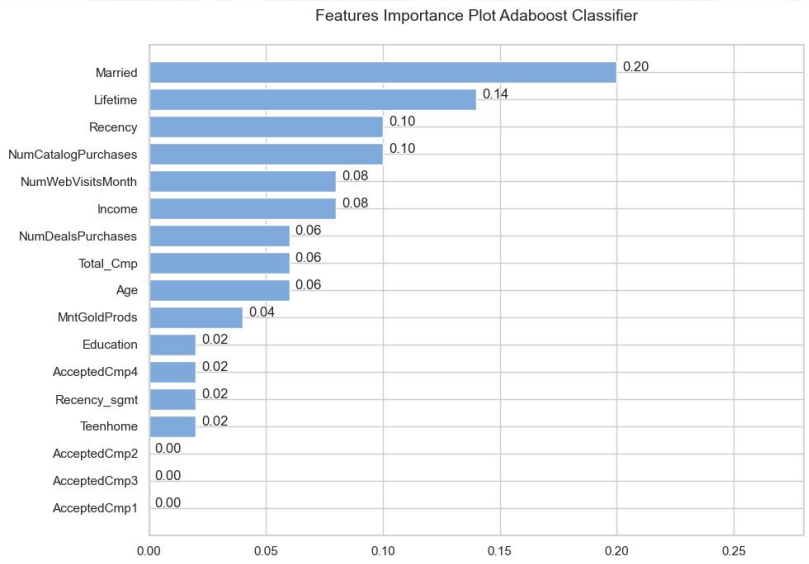
```
Training Accuracy: 87.3 %
Test Accuracy: 85.54 %
```

### Performance of Testing Model

```
Classification Report Testing Model (Adaboost Classifier):

Accuracy = 0.855
Precision = 0.511
Recall = 0.578
F1 Score = 0.542
Cross Val F1 (k=5) = 0.539
ROC AUC = 0.892
Cross Val ROC AUC (k=5) = 0.885

              precision    recall  f1-score   support

           0       0.92      0.90      0.91       477
           1       0.51      0.58      0.54        83

    accuracy                           0.86       560
   macro avg       0.72      0.74      0.73       560
weighted avg       0.86      0.86      0.86       560

==== Actual Data (Test) =====
Total = 560
No Response = 477
Response = 83
==== Predicted Data (Test) =====
TP = 48, FP = 46, TN = 431, FN = 35
Predictly Correct = 479
Predictly Wrong = 81
```



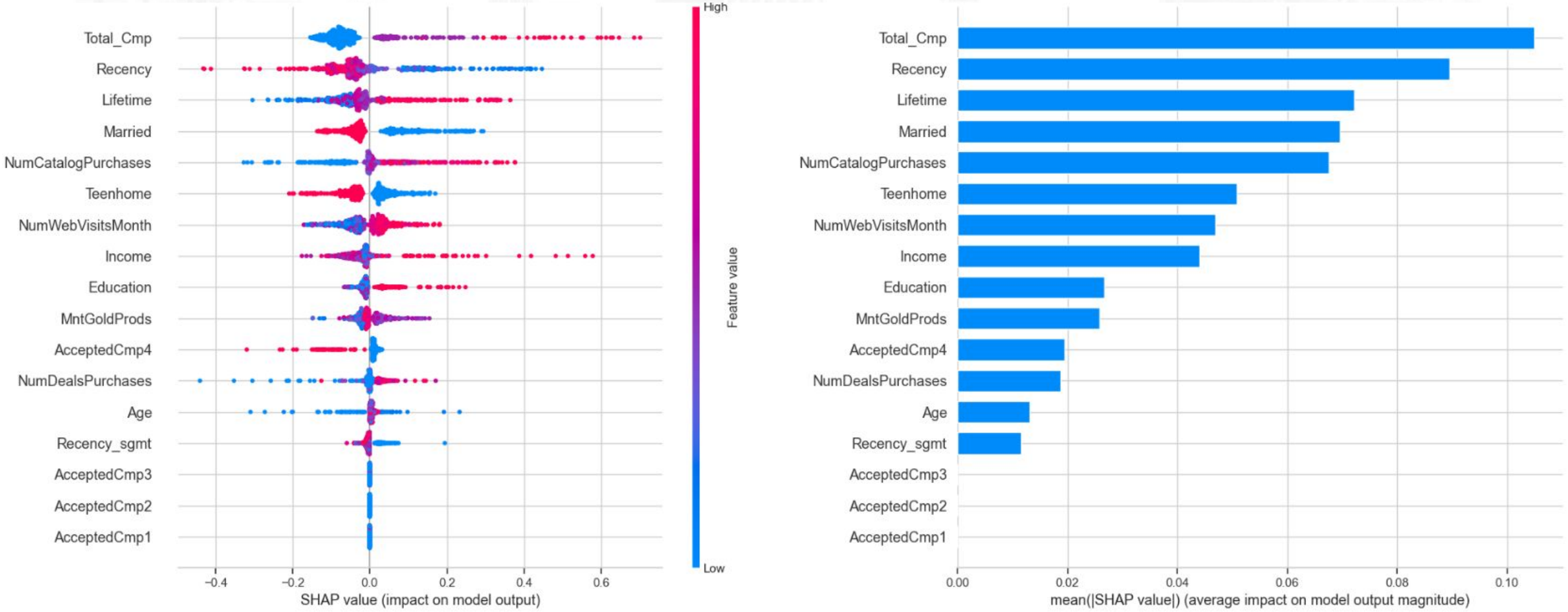



# Modelling

## 7. Adaboost Clasifier

### Performance of Testing Model



# Modelling

## 8. XGBoost Classifier

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.997000 | 0.889000 | 0.108000 |
| 1 | Precision | 0.996000 | 0.652000 | 0.344000 |
| 2 | Recall | 0.994000 | 0.542000 | 0.452000 |
| 3 | F1 Score | 0.995000 | 0.592000 | 0.403000 |
| 4 | F1 Score (crossval) | 0.972000 | 0.525000 | 0.447000 |
| 5 | ROC AUC | 1.000000 | 0.895000 | 0.105000 |
| 6 | ROC AUC (crossval) | 1.000000 | 0.887000 | 0.113000 |

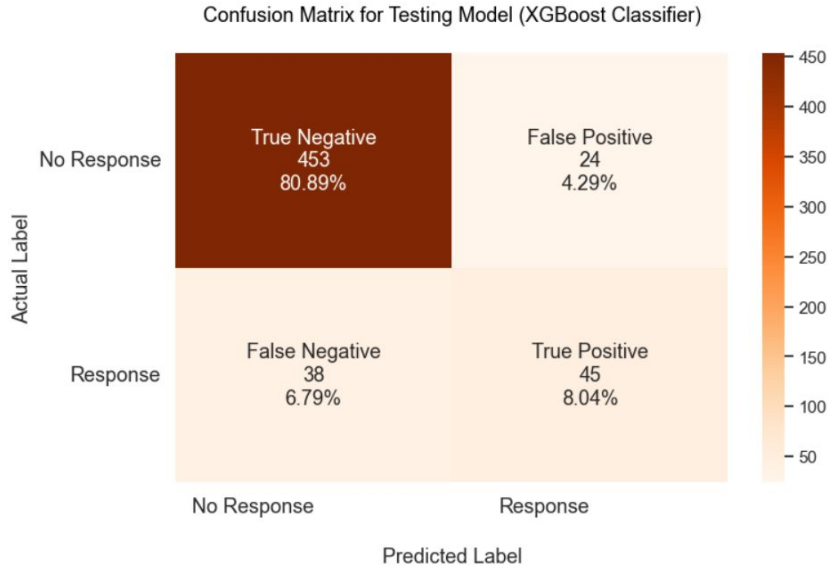
**Observation:**

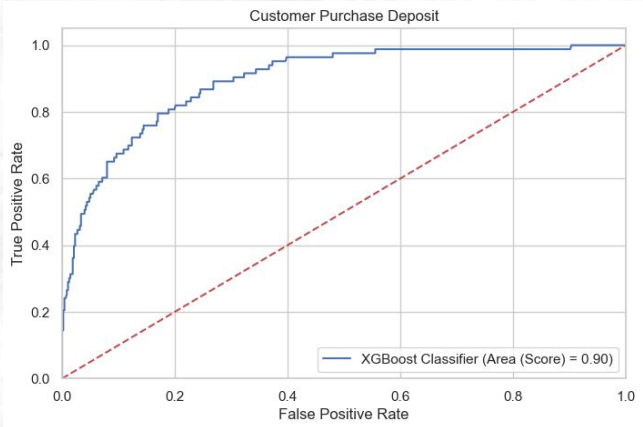
- Precision memiliki gap agak besar 0.34
- Recall memiliki gap cukup besar 0.45
- F1 Score memiliki gap cukup besar 0.40D

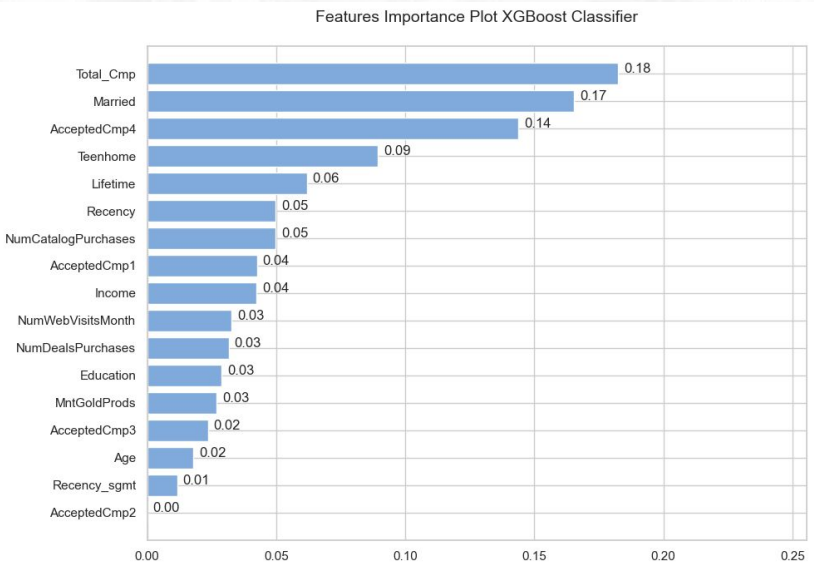
```
Training Accuracy: 99.67 %
Test Accuracy: 88.93 %
```

## Performance of Testing Model

```
Classification Report Testing Model (XGBoost Classifier

Accuracy = 0.889
Precision = 0.652
Recall = 0.542
F1 Score = 0.592
Cross Val F1 (k=5) = 0.525
ROC AUC = 0.895
Cross Val ROC AUC (k=5) = 0.887

              precision    recall  f1-score   support

           0       0.92      0.95      0.94       477
           1       0.65      0.54      0.59        83

    accuracy                           0.89       560
   macro avg       0.79      0.75      0.76       560
weighted avg       0.88      0.89      0.88       560

==== Actual Data (Test) =====
Total = 560
No Response = 477
Response = 83
==== Predicted Data (Test) =====
TP = 45, FP = 24, TN = 453, FN = 38
Predictly Correct = 498
Predictly Wrong = 62
```



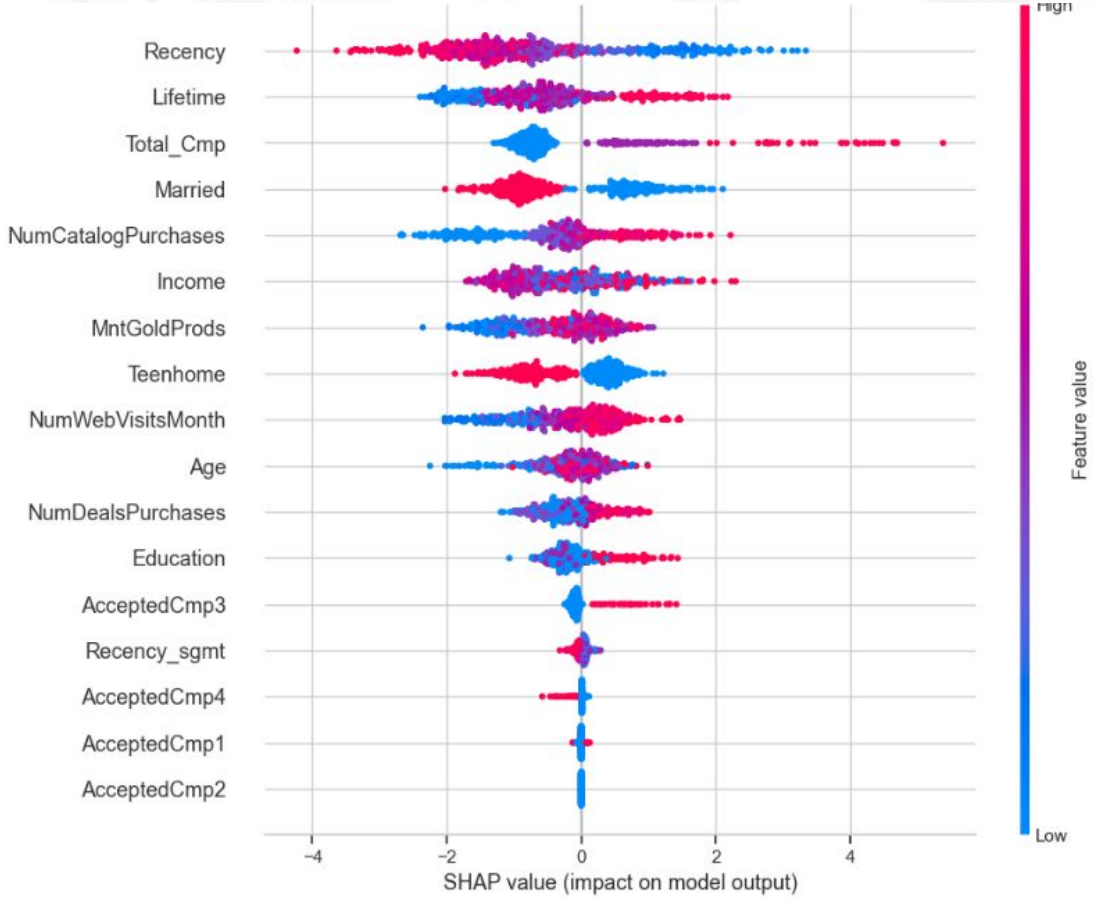



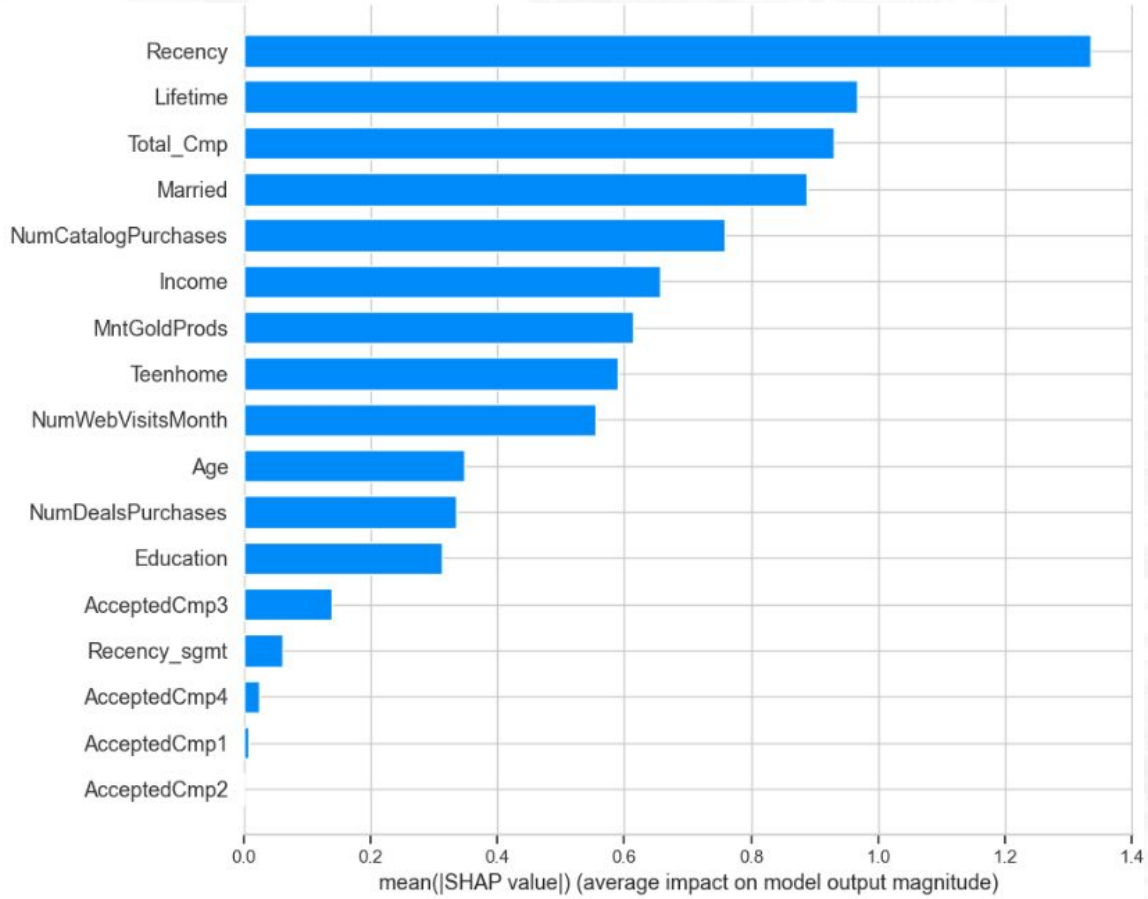



# Modelling

## 8. XGBoost Classifier

### Performance of Testing Model



# Modelling

## 9. Gradient Boosting Classifier

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.997000 | 0.846000 | 0.151000 |
| 1 | Precision | 1.000000 | 0.482000 | 0.518000 |
| 2 | Recall | 0.990000 | 0.494000 | 0.496000 |
| 3 | F1 Score | 0.995000 | 0.488000 | 0.507000 |
| 4 | F1 Score (crossval) | 0.972000 | 0.487000 | 0.485000 |
| 5 | ROC AUC | 1.000000 | 0.725000 | 0.275000 |
| 6 | ROC AUC (crossval) | 1.000000 | 0.752000 | 0.248000 |

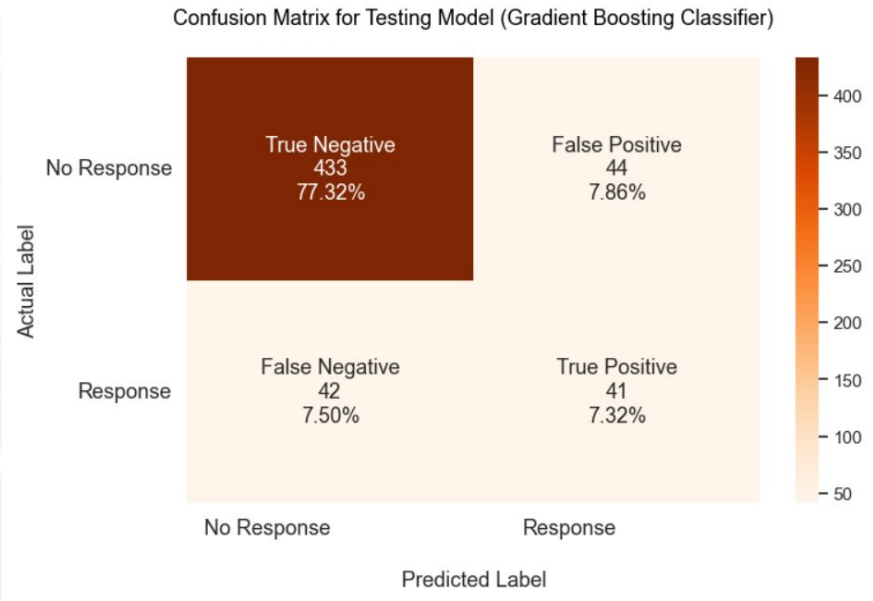
**Observation:**

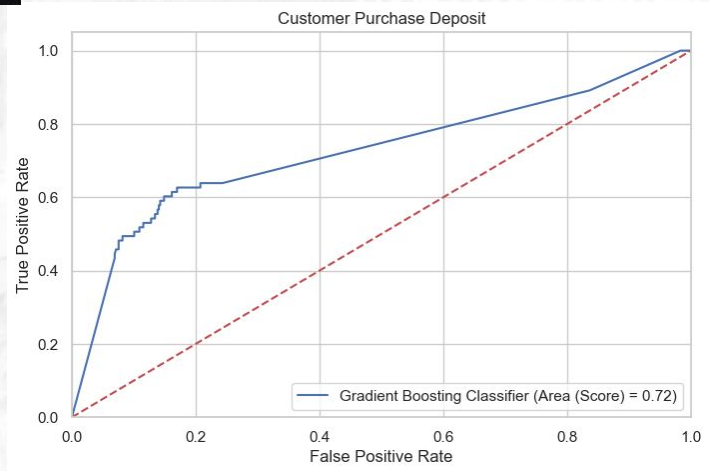
- Precision memiliki gap lumayan besar 0.51
- Recall memiliki gap cukup besar 0.49
- F1 Score memiliki gap cukup besar 0.50

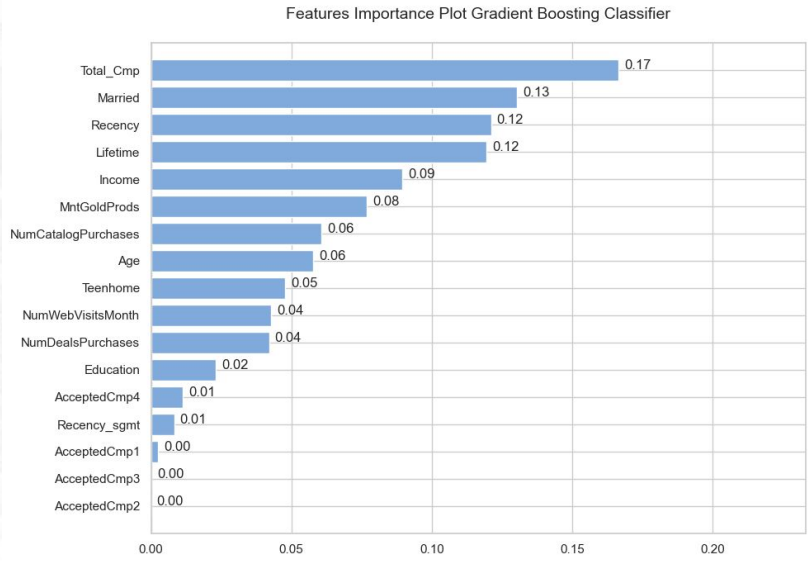
```
Training Accuracy: 99.67 %
Test Accuracy: 84.64 %
```

### Performance of Testing Model

```
Classification Report Testing Model (Gradient Boosting Classifier):

Accuracy = 0.846
Precision = 0.482
Recall = 0.494
F1 Score = 0.488
Cross Val F1 (k=5) = 0.487
ROC AUC = 0.725
Cross Val ROC AUC (k=5) = 0.752

              precision    recall  f1-score   support

           0       0.91      0.91      0.91       477
           1       0.48      0.49      0.49        83

    accuracy                           0.85       560
   macro avg       0.70      0.70      0.70       560
weighted avg       0.85      0.85      0.85       560

==== Actual Data (Test) =====
Total = 560
No Response = 477
Response = 83
==== Predicted Data (Test) =====
TP = 41, FP = 44, TN = 433, FN = 42
Predictly Correct = 474
Predictly Wrong = 86
```



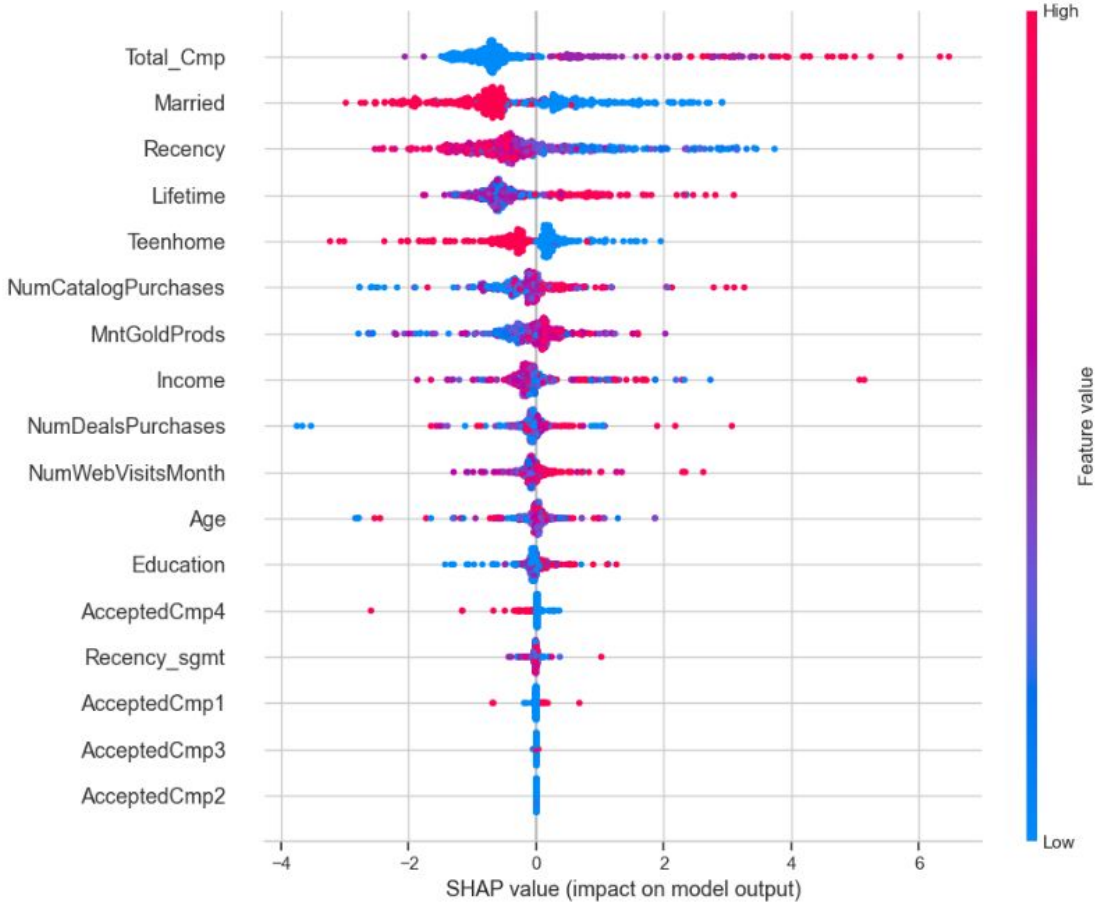



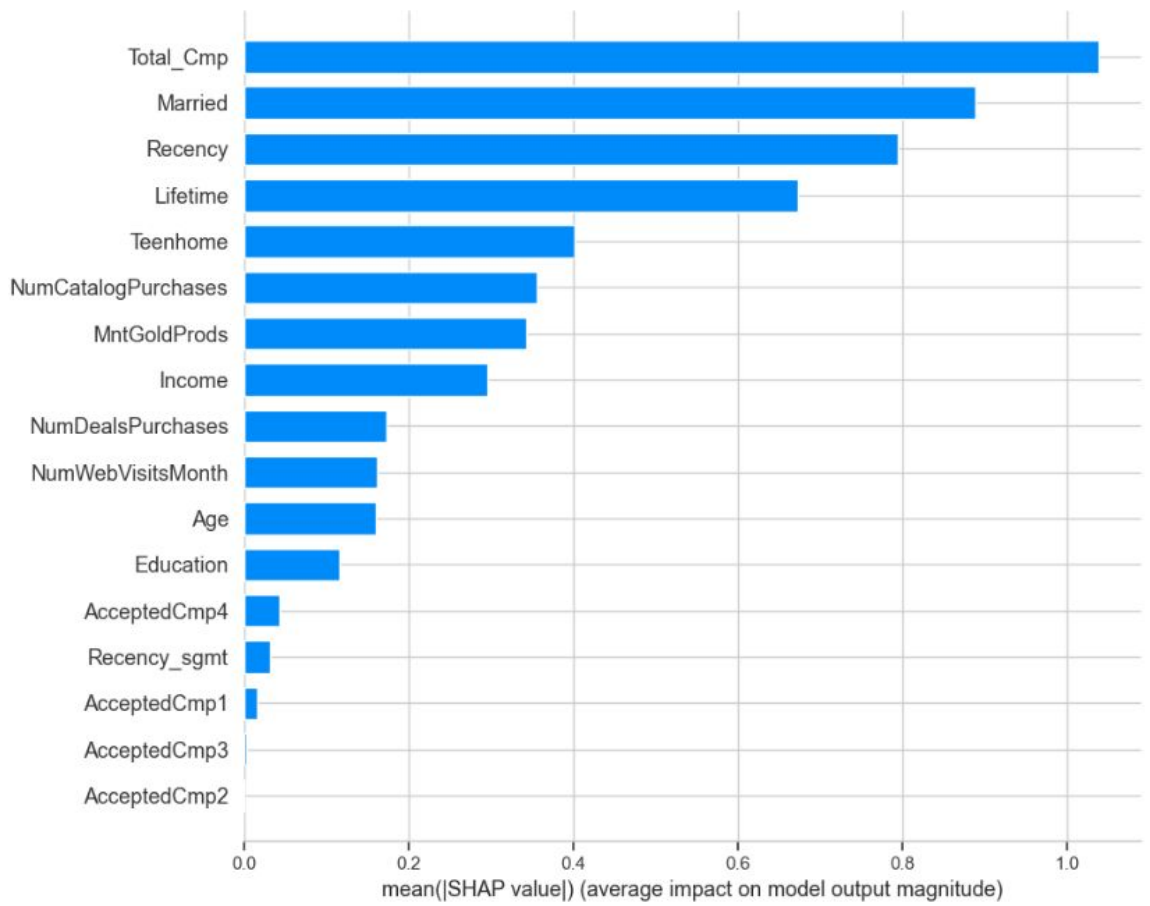



# Modelling

## 9. Gradient Boosting Classifier

### Performance of Testing Model



# Modelling

## 10. Support Vector Machine

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.879000 | 0.859000 | 0.020000 |
| 1 | Precision | 0.829000 | 0.520000 | 0.309000 |
| 2 | Recall | 0.803000 | 0.614000 | 0.189000 |
| 3 | F1 Score | 0.816000 | 0.564000 | 0.252000 |
| 4 | F1 Score (crossval) | 0.538000 | 0.448000 | 0.090000 |
| 5 | ROC AUC | 0.953000 | 0.895000 | 0.058000 |
| 6 | ROC AUC (crossval) | 0.928000 | 0.884000 | 0.044000 |

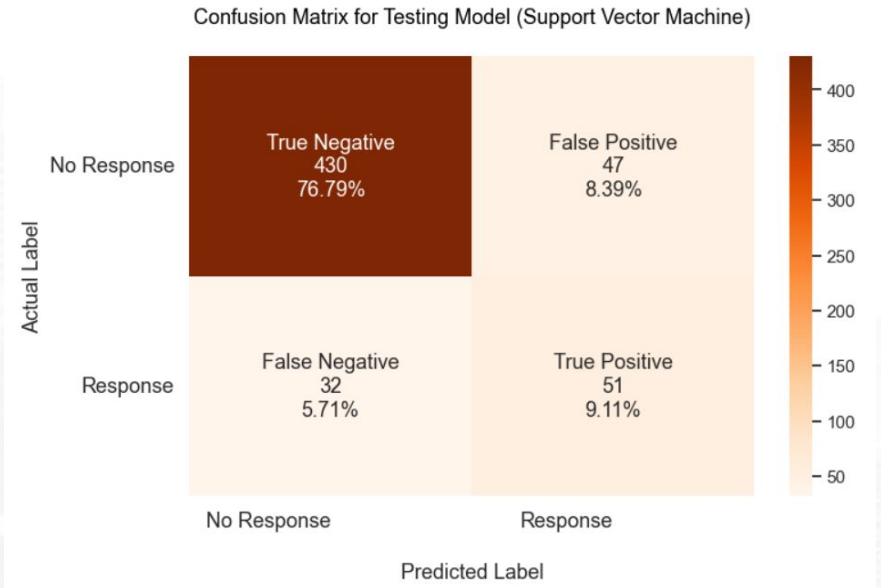
**Observation:**

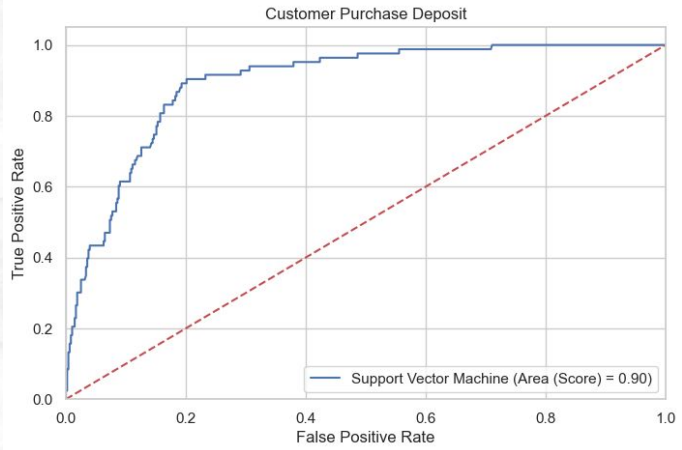
- Precision memiliki gap agak besar 0.30
- Recall memiliki gap cukup kecil 0.18
- F1 Score memiliki gap agak besar 0.25

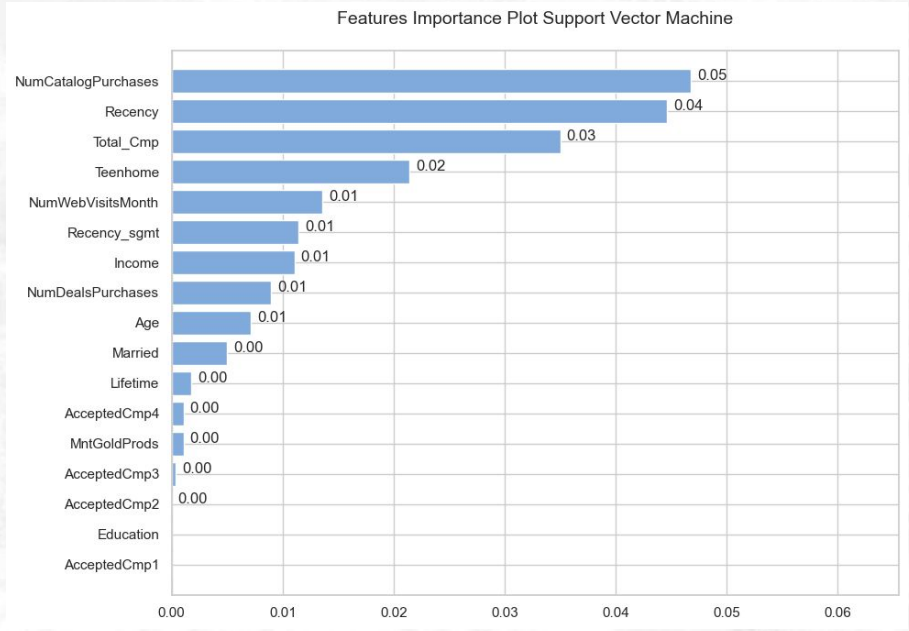
```
Training Accuracy: 87.91 %
Test Accuracy: 85.89 %
```

### Performance of Testing Model

```
Classification Report Testing Model (Support Vector Machine):

Accuracy = 0.859
Precision = 0.52
Recall = 0.614
F1 Score = 0.564
Cross Val F1 (k=5) = 0.448
ROC AUC = 0.895
Cross Val ROC AUC (k=5) = 0.884

              precision    recall  f1-score   support

           0       0.93      0.90      0.92       477
           1       0.52      0.61      0.56        83

    accuracy                           0.86       560
   macro avg       0.73      0.76      0.74       560
weighted avg       0.87      0.86      0.86       560

==== Actual Data (Test) =====
Total = 560
No Response = 477
Response = 83
==== Predicted Data (Test) =====
TP = 51, FP = 47, TN = 430, FN = 32
Predictly Correct = 481
Predictly Wrong = 79
```



# Modelling

## Summary

| | Models | Precision (Train) | Precision (Test) | Recall (Train) | Recall (Test) | F1 Score (Train) | F1 Score (Test) |
|---|---|---|---|---|---|---|---|
| 0 | Random Forest | 0.996000 | 0.786000 | 0.994000 | 0.530000 | 0.995000 | 0.633000 |
| 1 | XGBoost Classifier | 0.996000 | 0.652000 | 0.994000 | 0.542000 | 0.995000 | 0.592000 |
| 2 | Logistic Regression | 0.750000 | 0.523000 | 0.660000 | 0.675000 | 0.702000 | 0.589000 |
| 3 | Support Vector Machine | 0.829000 | 0.520000 | 0.803000 | 0.614000 | 0.816000 | 0.564000 |
| 4 | K-Nearest Neighbors | 0.819000 | 0.482000 | 0.959000 | 0.639000 | 0.884000 | 0.549000 |
| 5 | Adaboost Classifier | 0.844000 | 0.511000 | 0.759000 | 0.578000 | 0.799000 | 0.542000 |
| 6 | MLP Classifier | 0.992000 | 0.483000 | 0.993000 | 0.518000 | 0.992000 | 0.500000 |
| 7 | Gradient Boosting Classifier | 1.000000 | 0.482000 | 0.990000 | 0.494000 | 0.995000 | 0.488000 |
| 8 | Naive Bayes | 0.658000 | 0.441000 | 0.429000 | 0.542000 | 0.519000 | 0.486000 |
| 9 | Decision Tree | 1.000000 | 0.456000 | 0.990000 | 0.494000 | 0.995000 | 0.474000 |

Berdasarkan hasil modelling menggunakan beberapa algoritma di atas, dapat diketahui bahwa **nilai precision dan recall** pada dataset train dan test yang paling baik dihasilkan oleh Decision Tree, Random Forest, dan XGBoost dengan masing-masing nilai yang didapatkan sebagai berikut:

- **Random Forest:** Precision 0.841, Recall 0.446
- **XGBoost Classifier:** Precision 0.695, Recall 0.494
- **Logistic Regression:** Precision 0.729, Recall 0.422

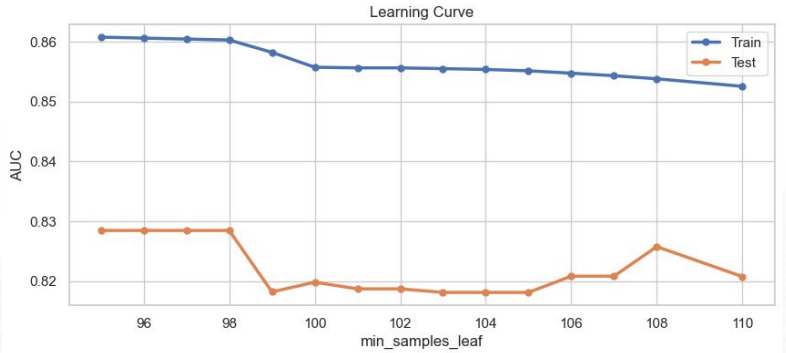
# Hyperparameter Tuning

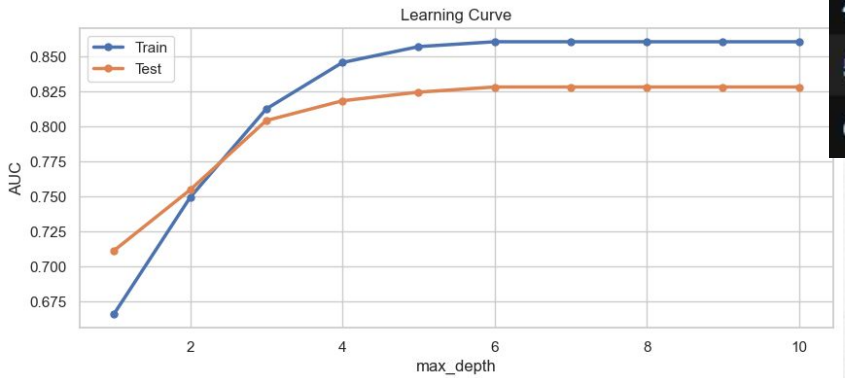
## 1. Decision Tree

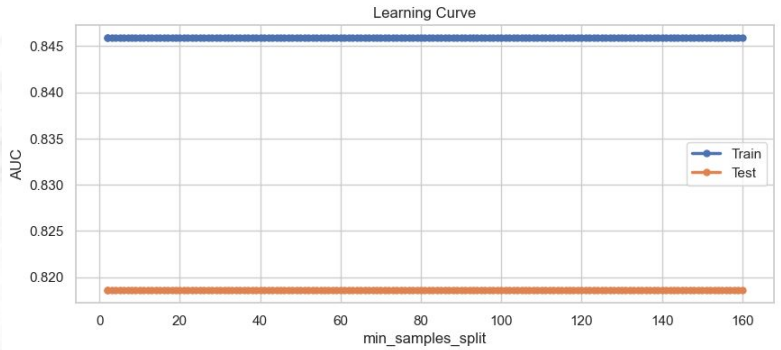
### Tuning CV

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.997000 | 0.846000 | 0.151000 |
| 1 | Precision | 1.000000 | 0.484000 | 0.516000 |
| 2 | Recall | 0.990000 | 0.554000 | 0.436000 |
| 3 | F1 Score | 0.995000 | 0.517000 | 0.478000 |
| 4 | F1 Score (crossval) | 0.972000 | 0.494000 | 0.478000 |
| 5 | ROC AUC | 1.000000 | 0.730000 | 0.270000 |
| 6 | ROC AUC (crossval) | 1.000000 | 0.713000 | 0.287000 |

### Learning Curve




### Manually Tuning

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.783000 | 0.830000 | -0.047000 |
| 1 | Precision | 0.719000 | 0.441000 | 0.278000 |
| 2 | Recall | 0.574000 | 0.542000 | 0.032000 |
| 3 | F1 Score | 0.639000 | 0.486000 | 0.153000 |
| 4 | F1 Score (crossval) | 0.414000 | 0.350000 | 0.064000 |
| 5 | ROC AUC | 0.846000 | 0.819000 | 0.027000 |
| 6 | ROC AUC (crossval) | 0.843000 | 0.799000 | 0.044000 |

Best value for min_samples_leaf = 95
Best value for max_depth = 4
Best value for min_sample_split = 20

# Hyperparameter Tuning

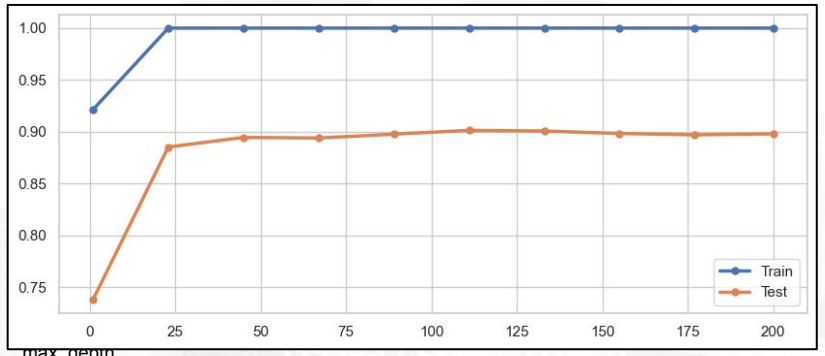
## 2. Random Forest

### Tuning CV

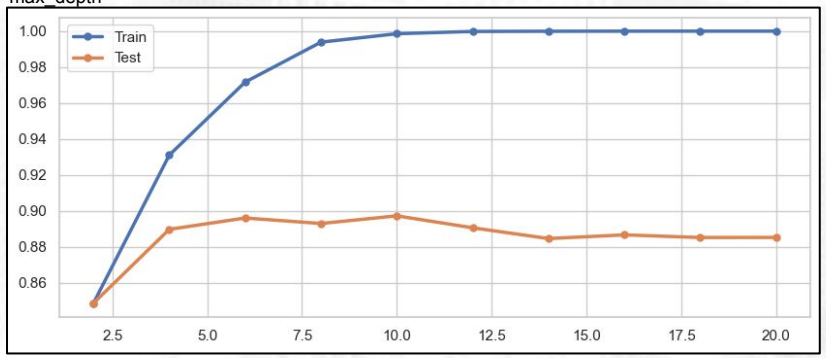
| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.997000 | 0.907000 | 0.090000 |
| 1 | Precision | 0.997000 | 0.772000 | 0.225000 |
| 2 | Recall | 0.993000 | 0.530000 | 0.463000 |
| 3 | F1 Score | 0.995000 | 0.629000 | 0.366000 |
| 4 | F1 Score (crossval) | 0.972000 | 0.533000 | 0.439000 |
| 5 | ROC AUC | 1.000000 | 0.898000 | 0.102000 |
| 6 | ROC AUC (crossval) | 1.000000 | 0.893000 | 0.107000 |

### Learning Curve

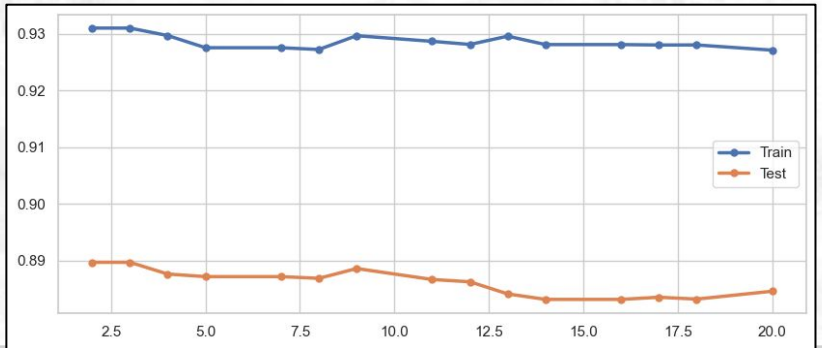
n_estimator



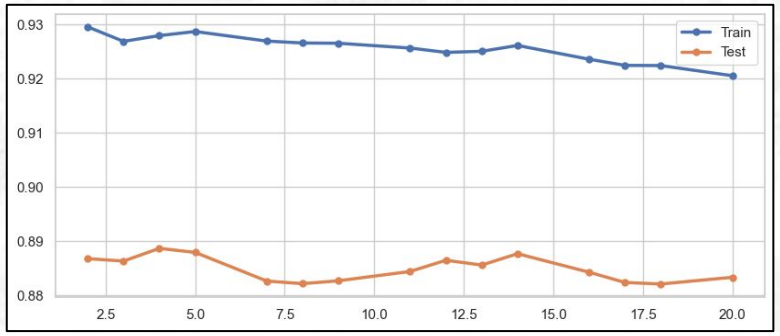

max_depth




min_samples_split




min_samples_leaf




### Manually Tuning

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.831000 | 0.893000 | -0.062000 |
| 1 | Precision | 0.867000 | 0.702000 | 0.165000 |
| 2 | Recall | 0.584000 | 0.482000 | 0.102000 |
| 3 | F1 Score | 0.698000 | 0.571000 | 0.127000 |
| 4 | F1 Score (crossval) | 0.439000 | 0.375000 | 0.064000 |
| 5 | ROC AUC | 0.926000 | 0.884000 | 0.042000 |
| 6 | ROC AUC (crossval) | 0.886000 | 0.858000 | 0.028000 |

Best value for n_estimator = 23
Best value for max_depth = 4
Best value for min_samples_split = 3
Best value for min_samples_leaf = 4

# Hyperparameter Tuning

## 3. Logistic Regression

### Tuning CV

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.815000 | 0.862000 | -0.047000 |
| 1 | Precision | 0.755000 | 0.528000 | 0.227000 |
| 2 | Recall | 0.660000 | 0.675000 | -0.015000 |
| 3 | F1 Score | 0.704000 | 0.593000 | 0.111000 |
| 4 | F1 Score (crossval) | 0.532000 | 0.497000 | 0.035000 |
| 5 | ROC AUC | 0.886000 | 0.901000 | -0.015000 |
| 6 | ROC AUC (crossval) | 0.891000 | 0.885000 | 0.006000 |

**Best using {'penalty': 'l2', 'C': 0.4747999999999994}**

## 4. K-Nearest Neighbors

### Tuning CV

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.867000 | 0.838000 | 0.029000 |
| 1 | Precision | 0.763000 | 0.465000 | 0.298000 |
| 2 | Recall | 0.871000 | 0.639000 | 0.232000 |
| 3 | F1 Score | 0.814000 | 0.538000 | 0.276000 |
| 4 | F1 Score (crossval) | 0.411000 | 0.334000 | 0.077000 |
| 5 | ROC AUC | 0.949000 | 0.857000 | 0.092000 |
| 6 | ROC AUC (crossval) | 0.921000 | 0.848000 | 0.073000 |

**Best using {'p': 2, 'n_neighbors': 13, 'leaf_size': 34, 'algorithm': 'brute'}**

# Hyperparameter Tuning

## 4. Adaboost Classifier

### Tuning CV

```
# List of hyperparameter
hyperparameters = dict(n_estimators = [int(x) for x in np.linspace(start = 50, stop = 2000, num = 2000)], # Jumlah iterasi
                       learning_rate = [float(x) for x in np.linspace(start = 0.001, stop = 0.1, num = 200)],
                       algorithm = ['SAMME', 'SAMME.R']
                      )

adab = AdaBoostClassifier(random_state=42)
cv = StratifiedKFold(n_splits=5)

adab_model_ht = RandomizedSearchCV(adab, hyperparameters, cv=cv, scoring=custom_scoring, refit="f1", random_state=1)
# adab_model_ht = GridSearchCV(adab, hyperparameters, cv=cv, scoring=custom_scoring, refit="f1")

grid_result = adab_model_ht.fit(X_train, y_train)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
# means = grid_result.cv_results_['mean_test_f1']
# stds = grid_result.cv_results_['std_test_f1']
# params = grid_result.cv_results_['params']
# for mean, stdev, param in zip(means, stds, params):
#     print("%f (%f) with: %r" % (mean, stdev, param))
```

```
Best: 0.764393 using {'n_estimators': 1634, 'learning_rate': 0.09801005025125628, 'algorithm': 'SAMME.R'}
```

Best: 0.521426 using {'n_estimators': 1634, 'learning_rate': 0.09801005025125628, 'algorithm': 'SAMME.R'}

```
adab_model_htt = AdaBoostClassifier(n_estimators = 1634, learning_rate = 0.09801005025125628, algorithm = 'SAMME.R', random_state=42)
adab_model_htt.fit(X_train,y_train)
```

```
AdaBoostClassifier(learning_rate=0.09801005025125628, n_estimators=1634,
                   random_state=42)
```

# Hyperparameter Tuning

## 5. XGBoost Classifier

### Tuning CV

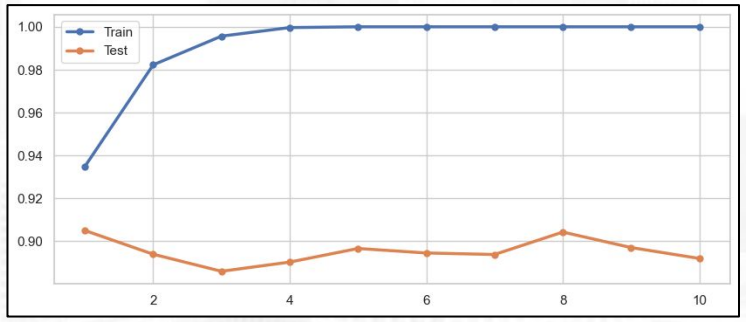
| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.966000 | 0.898000 | 0.068000 |
| 1 | Precision | 0.976000 | 0.710000 | 0.266000 |
| 2 | Recall | 0.920000 | 0.530000 | 0.390000 |
| 3 | F1 Score | 0.947000 | 0.607000 | 0.340000 |
| 4 | F1 Score (crossval) | 0.759000 | 0.522000 | 0.237000 |
| 5 | ROC AUC | 0.994000 | 0.894000 | 0.100000 |
| 6 | ROC AUC (crossval) | 0.975000 | 0.889000 | 0.086000 |

### Manually Tuning

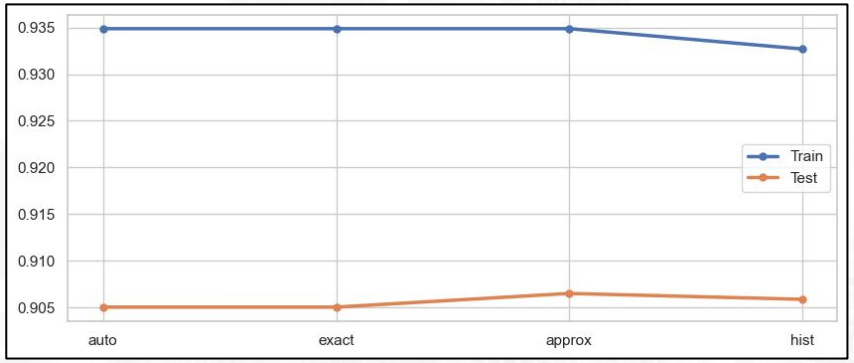
| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.851000 | 0.879000 | -0.028000 |
| 1 | Precision | 0.826000 | 0.595000 | 0.231000 |
| 2 | Recall | 0.699000 | 0.566000 | 0.133000 |
| 3 | F1 Score | 0.757000 | 0.580000 | 0.177000 |
| 4 | F1 Score (crossval) | 0.560000 | 0.519000 | 0.041000 |
| 5 | ROC AUC | 0.931000 | 0.905000 | 0.026000 |
| 6 | ROC AUC (crossval) | 0.911000 | 0.889000 | 0.022000 |

### Learning Curve

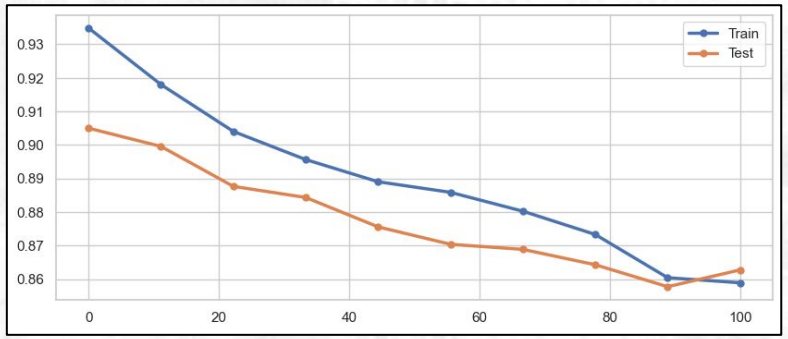
**max_depth =>** best value = 1



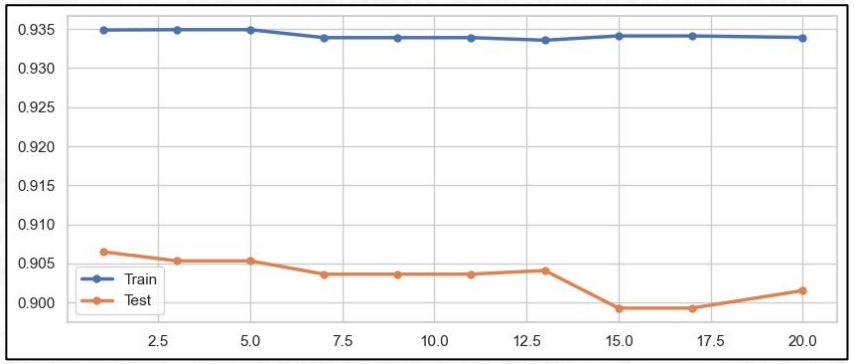

**tree_method =>** best value = approx




**gamma =>** best value = 0




**min_child_weight =>** best value =5



# Model Additional

## Stacking

```
Model: LogisticRegression, Score: 0.7862519259427647
Model: KNeighborsClassifier, Score: 0.7346492889798186
Model: DecisionTreeClassifier, Score: 0.6209744638933091
Model: RandomForestClassifier, Score: 0.760336945265338
Model: AdaBoostClassifier, Score: 0.7511555656588619
Model: XGBClassifier, Score: 0.7126619686292339
```

## Stacking Implementation

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.989000 | 0.868000 | 0.121000 |
| 1 | Precision | 0.975000 | 0.562000 | 0.413000 |
| 2 | Recall | 0.993000 | 0.494000 | 0.499000 |
| 3 | F1 Score | 0.984000 | 0.526000 | 0.458000 |
| 4 | F1 Score (crossval) | 0.742000 | 0.514000 | 0.228000 |
| 5 | ROC AUC | 1.000000 | 0.880000 | 0.120000 |
| 6 | ROC AUC (crossval) | 0.962000 | 0.871000 | 0.091000 |

## Voting Classifier

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.997000 | 0.898000 | 0.099000 |
| 1 | Precision | 0.997000 | 0.681000 | 0.316000 |
| 2 | Recall | 0.993000 | 0.590000 | 0.403000 |
| 3 | F1 Score | 0.995000 | 0.632000 | 0.363000 |
| 4 | F1 Score (crossval) | 0.969000 | 0.529000 | 0.440000 |
| 5 | ROC AUC | 1.000000 | 0.910000 | 0.090000 |
| 6 | ROC AUC (crossval) | 1.000000 | 0.892000 | 0.108000 |

# Model Comparison

**Perbandingan precision, recall dan f1 score pada train test tertinggi**

- Tanpa Tuning

| | Models | Precision (Train) | Precision (Test) | Recall (Train) | Recall (Test) | F1 Score (Train) | F1 Score (Test) |
|---|---|---|---|---|---|---|---|
| 0 | Random Forest | 0.996 | 0.786 | 0.994 | 0.530 | 0.995 | 0.633 |
| 1 | Voting Classifier | 0.997 | 0.681 | 0.993 | 0.590 | 0.995 | 0.632 |
| 2 | XGBoost Classifier | 0.996 | 0.652 | 0.994 | 0.542 | 0.995 | 0.592 |
| 3 | Logistic Regression | 0.750 | 0.523 | 0.660 | 0.675 | 0.702 | 0.589 |
| 4 | Support Vector Machine | 0.829 | 0.520 | 0.803 | 0.614 | 0.816 | 0.564 |
| 5 | K-Nearest Neighbors | 0.819 | 0.482 | 0.959 | 0.639 | 0.884 | 0.549 |
| 6 | Adaboost Classifier | 0.844 | 0.511 | 0.759 | 0.578 | 0.799 | 0.542 |
| 7 | Stacking Classifier | 0.975 | 0.562 | 0.993 | 0.494 | 0.984 | 0.526 |
| 8 | MLP Classifier | 0.992 | 0.483 | 0.993 | 0.518 | 0.992 | 0.500 |
| 9 | Gradient Boosting Classifier | 1.000 | 0.482 | 0.990 | 0.494 | 0.995 | 0.488 |
| 10 | Naive Bayes | 0.658 | 0.441 | 0.429 | 0.542 | 0.519 | 0.486 |
| 11 | Decision Tree | 1.000 | 0.456 | 0.990 | 0.494 | 0.995 | 0.474 |

Melihat summary metrics dari berbagai algoritma yang telah dibuat
* HT --> Hyperparameter Tuning
* HT2 --> Manually Tuning

- Dengan Tuning

| | Models | Precision (Train) | Precision (Test) | Recall (Train) | Recall (Test) | F1 Score (Train) | F1 Score (Test) |
|---|---|---|---|---|---|---|---|
| 0 | Random Forest (HT1) | 0.997 | 0.772 | 0.993 | 0.530 | 0.995 | 0.629 |
| 1 | XGBoost Classifier (HT) | 0.976 | 0.710 | 0.920 | 0.530 | 0.947 | 0.607 |
| 2 | Random Forest (HT2) | 0.972 | 0.724 | 0.912 | 0.506 | 0.941 | 0.596 |
| 3 | Logistic Regression (HT) | 0.755 | 0.528 | 0.660 | 0.675 | 0.704 | 0.593 |
| 4 | XGBoost Classifier (HT2) | 0.826 | 0.595 | 0.699 | 0.566 | 0.757 | 0.580 |
| 5 | Random Forest (HT3) | 0.867 | 0.702 | 0.584 | 0.482 | 0.698 | 0.571 |
| 6 | K-Nearest Neighbors (HT) | 0.763 | 0.465 | 0.871 | 0.639 | 0.814 | 0.538 |
| 7 | Decision Tree (HT) | 1.000 | 0.484 | 0.990 | 0.554 | 0.995 | 0.517 |
| 8 | Decision Tree (HT2) | 0.719 | 0.441 | 0.574 | 0.542 | 0.639 | 0.486 |

# Model Comparison

**Perbandingan precision, recall dan f1 score pada train test dengan Total Difference terendah**

- Tanpa Tuning

| | Models | Precision (Train) | Precision (Test) | Recall (Train) | Recall (Test) | F1 Score (Train) | F1 Score (Test) | Total Diff |
|---|---|---|---|---|---|---|---|---|
| 0 | Naive Bayes | 0.658 | 0.441 | 0.429 | 0.542 | 0.519 | 0.486 | 0.137 |
| 1 | Logistic Regression | 0.750 | 0.523 | 0.660 | 0.675 | 0.702 | 0.589 | 0.325 |
| 2 | Support Vector Machine | 0.829 | 0.520 | 0.803 | 0.614 | 0.816 | 0.564 | 0.750 |
| 3 | Adaboost Classifier | 0.844 | 0.511 | 0.759 | 0.578 | 0.799 | 0.542 | 0.771 |
| 4 | K-Nearest Neighbors | 0.819 | 0.482 | 0.959 | 0.639 | 0.884 | 0.549 | 0.992 |
| 5 | Random Forest | 0.996 | 0.786 | 0.994 | 0.530 | 0.995 | 0.633 | 1.036 |
| 6 | Voting Classifier | 0.997 | 0.681 | 0.993 | 0.590 | 0.995 | 0.632 | 1.082 |
| 7 | XGBoost Classifier | 0.996 | 0.652 | 0.994 | 0.542 | 0.995 | 0.592 | 1.199 |
| 8 | Stacking Classifier | 0.975 | 0.562 | 0.993 | 0.494 | 0.984 | 0.526 | 1.370 |
| 9 | MLP Classifier | 0.992 | 0.483 | 0.993 | 0.518 | 0.992 | 0.500 | 1.476 |
| 10 | Gradient Boosting Classifier | 1.000 | 0.482 | 0.990 | 0.494 | 0.995 | 0.488 | 1.521 |
| 11 | Decision Tree | 1.000 | 0.456 | 0.990 | 0.494 | 0.995 | 0.474 | 1.561 |

Melihat summary metrics dari berbagai algoritma yang telah dibuat
* HT --> Hyperparameter Tuning
* HT2 --> Manually Tuning

- Dengan Tuning

| | Models | Precision (Train) | Precision (Test) | Recall (Train) | Recall (Test) | F1 Score (Train) | F1 Score (Test) | Total Diff |
|---|---|---|---|---|---|---|---|---|
| 0 | Logistic Regression (HT) | 0.755 | 0.528 | 0.660 | 0.675 | 0.704 | 0.593 | 0.323 |
| 1 | Random Forest (HT3) | 0.867 | 0.702 | 0.584 | 0.482 | 0.698 | 0.571 | 0.394 |
| 2 | Decision Tree (HT2) | 0.719 | 0.441 | 0.574 | 0.542 | 0.639 | 0.486 | 0.463 |
| 3 | XGBoost Classifier (HT2) | 0.826 | 0.595 | 0.699 | 0.566 | 0.757 | 0.580 | 0.541 |
| 4 | K-Nearest Neighbors (HT) | 0.763 | 0.465 | 0.871 | 0.639 | 0.814 | 0.538 | 0.806 |
| 5 | XGBoost Classifier (HT) | 0.976 | 0.710 | 0.920 | 0.530 | 0.947 | 0.607 | 0.996 |
| 6 | Random Forest (HT2) | 0.972 | 0.724 | 0.912 | 0.506 | 0.941 | 0.596 | 0.999 |
| 7 | Random Forest (HT1) | 0.997 | 0.772 | 0.993 | 0.530 | 0.995 | 0.629 | 1.054 |
| 8 | Decision Tree (HT) | 1.000 | 0.484 | 0.990 | 0.554 | 0.995 | 0.517 | 1.430 |

# Model Comparison

Rakamin Academy

## Evaluation Metrics pada Train Urutan Tertinggi

| | Model (Train) | Accuracy | Precision | Recall | F1 Score | Cross Val F1 (k=5) | ROC AUC | Cross Val ROC AUC (k=5) |
|---|---|---|---|---|---|---|---|---|
| 0 | Decision Tree | 0.997 | 1.000 | 0.990 | 0.995 | 0.972 | 1.000 | 1.000 |
| 1 | Gradient Boosting Classifier | 0.997 | 1.000 | 0.990 | 0.995 | 0.972 | 1.000 | 1.000 |
| 2 | Decision Tree (HT) | 0.997 | 1.000 | 0.990 | 0.995 | 0.972 | 1.000 | 1.000 |
| 3 | Random Forest (HT1) | 0.997 | 0.997 | 0.993 | 0.995 | 0.972 | 1.000 | 1.000 |
| 4 | Voting Classifier | 0.997 | 0.997 | 0.993 | 0.995 | 0.969 | 1.000 | 1.000 |
| 5 | Random Forest | 0.997 | 0.996 | 0.994 | 0.995 | 0.972 | 1.000 | 1.000 |
| 6 | XGBoost Classifier | 0.997 | 0.996 | 0.994 | 0.995 | 0.972 | 1.000 | 1.000 |
| 7 | MLP Classifier | 0.995 | 0.992 | 0.993 | 0.992 | 0.874 | 1.000 | 0.990 |
| 8 | Stacking Classifier | 0.989 | 0.975 | 0.993 | 0.984 | 0.742 | 1.000 | 0.962 |
| 9 | XGBoost Classifier (HT) | 0.966 | 0.976 | 0.920 | 0.947 | 0.759 | 0.994 | 0.975 |
| 10 | Random Forest (HT2) | 0.962 | 0.972 | 0.912 | 0.941 | 0.638 | 0.993 | 0.978 |
| 11 | K-Nearest Neighbors | 0.916 | 0.819 | 0.959 | 0.884 | 0.605 | 0.982 | 0.949 |
| 12 | Support Vector Machine | 0.879 | 0.829 | 0.803 | 0.816 | 0.538 | 0.953 | 0.928 |
| 13 | K-Nearest Neighbors (HT) | 0.867 | 0.763 | 0.871 | 0.814 | 0.411 | 0.949 | 0.921 |
| 14 | Adaboost Classifier | 0.873 | 0.844 | 0.759 | 0.799 | 0.605 | 0.948 | 0.929 |
| 15 | XGBoost Classifier (HT2) | 0.851 | 0.826 | 0.699 | 0.757 | 0.560 | 0.931 | 0.911 |
| 16 | Logistic Regression (HT) | 0.815 | 0.755 | 0.660 | 0.704 | 0.532 | 0.886 | 0.891 |
| 17 | Logistic Regression | 0.813 | 0.750 | 0.660 | 0.702 | 0.536 | 0.887 | 0.891 |
| 18 | Random Forest (HT3) | 0.831 | 0.867 | 0.584 | 0.698 | 0.439 | 0.926 | 0.886 |
| 19 | Decision Tree (HT2) | 0.783 | 0.719 | 0.574 | 0.639 | 0.414 | 0.846 | 0.843 |
| 20 | Naive Bayes | 0.735 | 0.658 | 0.429 | 0.519 | 0.469 | 0.807 | 0.828 |

Melihat summary metrics dari berbagai algoritma yang telah dibuat
* HT --> Hyperparameter Tuning
* HT2 --> Manually Tuning

# Model Comparison

Rakamin Academy

## Evaluation Metrics pada Test Urutan Tertinggi

| | Model (Test) | Accuracy | Precision | Recall | F1 Score | Cross Val F1 (k=5) | ROC AUC | Cross Val ROC AUC (k=5) |
|---|---|---|---|---|---|---|---|---|
| 0 | Random Forest | 0.909 | 0.786 | 0.530 | 0.633 | 0.519 | 0.905 | 0.896 |
| 1 | Voting Classifier | 0.898 | 0.681 | 0.590 | 0.632 | 0.529 | 0.910 | 0.892 |
| 2 | Random Forest (HT1) | 0.907 | 0.772 | 0.530 | 0.629 | 0.533 | 0.898 | 0.893 |
| 3 | XGBoost Classifier (HT) | 0.898 | 0.710 | 0.530 | 0.607 | 0.522 | 0.894 | 0.889 |
| 4 | Random Forest (HT2) | 0.898 | 0.724 | 0.506 | 0.596 | 0.459 | 0.907 | 0.894 |
| 5 | Logistic Regression (HT) | 0.862 | 0.528 | 0.675 | 0.593 | 0.497 | 0.901 | 0.885 |
| 6 | XGBoost Classifier | 0.889 | 0.652 | 0.542 | 0.592 | 0.525 | 0.895 | 0.887 |
| 7 | Logistic Regression | 0.861 | 0.523 | 0.675 | 0.589 | 0.510 | 0.900 | 0.885 |
| 8 | XGBoost Classifier (HT2) | 0.879 | 0.595 | 0.566 | 0.580 | 0.519 | 0.905 | 0.889 |
| 9 | Random Forest (HT3) | 0.893 | 0.702 | 0.482 | 0.571 | 0.375 | 0.884 | 0.858 |
| 10 | Support Vector Machine | 0.859 | 0.520 | 0.614 | 0.564 | 0.448 | 0.895 | 0.884 |
| 11 | K-Nearest Neighbors | 0.845 | 0.482 | 0.639 | 0.549 | 0.400 | 0.842 | 0.810 |
| 12 | Adaboost Classifier | 0.855 | 0.511 | 0.578 | 0.542 | 0.539 | 0.892 | 0.885 |
| 13 | K-Nearest Neighbors (HT) | 0.838 | 0.465 | 0.639 | 0.538 | 0.334 | 0.857 | 0.848 |
| 14 | Stacking Classifier | 0.868 | 0.562 | 0.494 | 0.526 | 0.514 | 0.880 | 0.871 |
| 15 | Decision Tree (HT) | 0.846 | 0.484 | 0.554 | 0.517 | 0.494 | 0.730 | 0.713 |
| 16 | MLP Classifier | 0.846 | 0.483 | 0.518 | 0.500 | 0.558 | 0.830 | 0.887 |
| 17 | Gradient Boosting Classifier | 0.846 | 0.482 | 0.494 | 0.488 | 0.487 | 0.725 | 0.752 |
| 18 | Naive Bayes | 0.830 | 0.441 | 0.542 | 0.486 | 0.451 | 0.838 | 0.824 |
| 19 | Decision Tree (HT2) | 0.830 | 0.441 | 0.542 | 0.486 | 0.350 | 0.819 | 0.799 |
| 20 | Decision Tree | 0.838 | 0.456 | 0.494 | 0.474 | 0.488 | 0.699 | 0.718 |

Melihat summary metrics dari berbagai algoritma yang telah dibuat
* HT --> Hyperparameter Tuning
* HT2 --> Manually Tuning

# Model Selection

| | Evaluation Metrics | Train | Test | Diff Range |
|---|---|---|---|---|
| 0 | Accuracy | 0.997000 | 0.909000 | 0.088000 |
| 1 | Precision | 0.996000 | 0.786000 | 0.210000 |
| 2 | Recall | 0.994000 | 0.530000 | 0.464000 |
| 3 | F1 Score | 0.995000 | 0.633000 | 0.362000 |
| 4 | F1 Score (crossval) | 0.972000 | 0.519000 | 0.453000 |
| 5 | ROC AUC | 1.000000 | 0.905000 | 0.095000 |
| 6 | ROC AUC (crossval) | 1.000000 | 0.896000 | 0.104000 |

Dari ketiga model di atas, kami membandingkan nilai metric precision, recall, serta gap antara nilai train dan test pada masing-masing metric. Dari hasil perbandingan tersebut, kami dapatkan bahwasanya **model yang paling optimal untuk studi kasus kami adalah Random Forest**

# Model Comparison

Berdasarkan metode test modelling yang telah dilakukan, dapat disimpulkan bahwa model **Random Forest (HT)** memiliki hasil ***Precision, Recall, dan F1 Score yang paling stabil dan lumayan besar.***

## ✅ Random Forest (HT) ✅

***RandomForestClassifier(max_depth = 15, min_samples_leaf = 1, min_samples_split = 2, n_estimators = 300, random_state=42)***

- **Precision** Train : 0.996, Test : 0.786 (gap : 0.210)
- **Recall** Train : 0.994, Test : 0.530 (gap : 0.464)
- **F1 Score** Train : 0.995, Test : 0.633 (gap : 0.362)
- **F1 Score (Cross Validation)** Train : 0.972, Test : 0.519 (gap : 0.453)

**Total Different/Gap Train vs Test = 1.036**

- **Accuracy** Train : 0.997, Test : 0.909 (gap : 0.088)
- **ROC AUC** Train : 1.000, Test : 0.905 (gap : 0.095)
- **ROC AUC (Cross Validation)** Train : 1.000, Test : 0.896 (gap : 0.104)

Sehingga untuk selanjutnya, di tahap ***Feature Importance*** akan menggunakan metode ***Random Forest*** sebagai model terbaik untuk menemukan feature importance beserta **rekomendasi bisnis.**

# Model Selection

# Business Insight & Recommendation

# Model Selection

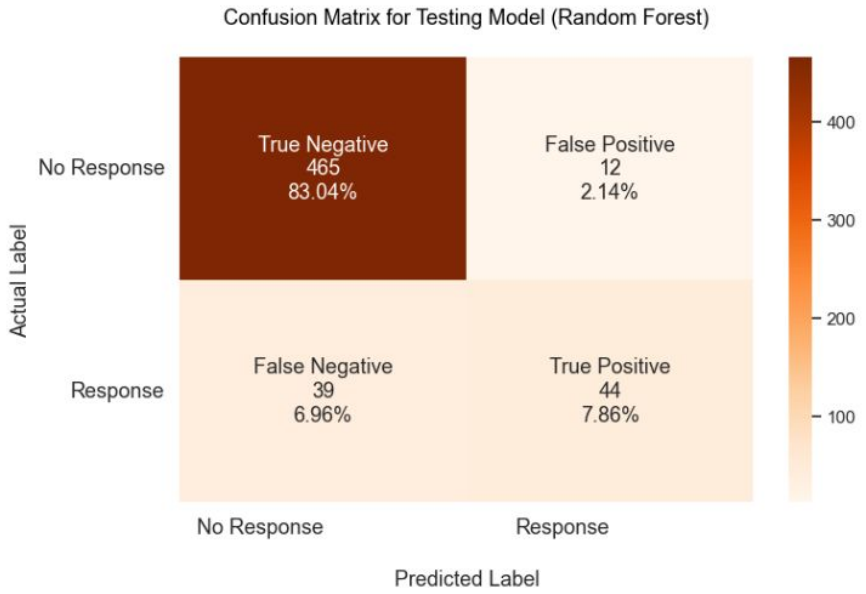
**Classification Report Testing Model (Random Forest (HT)):**

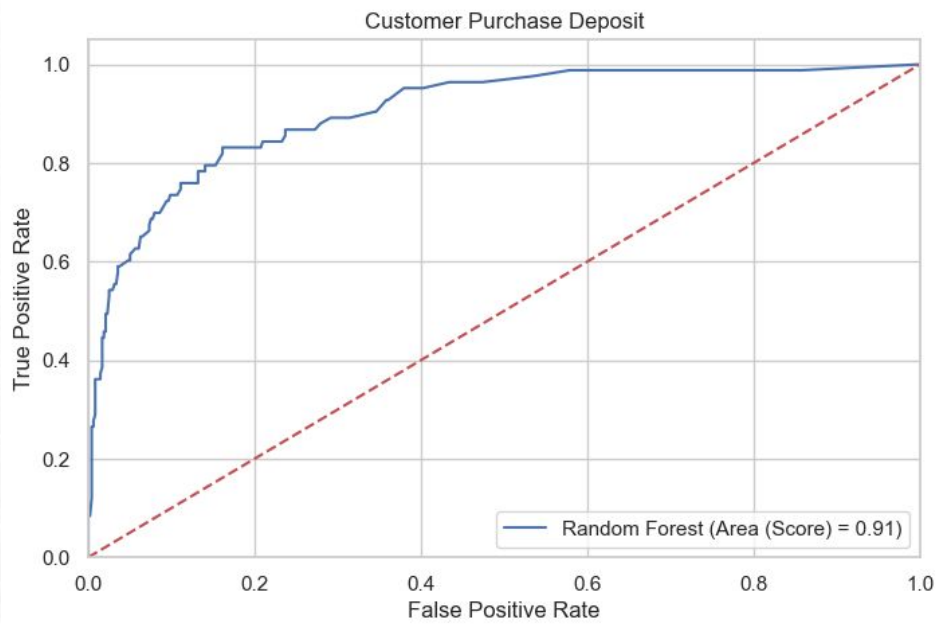
**Accuracy =** 0.909
**Precision =** 0.786
**Recall =** 0.53
**F1 Score =** 0.633
**Cross Val F1 (k=5) =** 0.519
**ROC AUC =** 0.905
**Cross Val ROC AUC (k=5) =** 0.896

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.97 | 0.95 | 477 |
| 1 | 0.79 | 0.53 | 0.63 | 83 |
| accuracy | | | 0.90 | 560 |
| macro avg | 0.85 | 0.75 | 0.79 | 560 |
| weighted avg | 0.90 | 0.91 | 0.90 | 560 |

**==== Actual Data (Test) =====**
Total = 560
No Response = 477
Response = 83
**==== Predicted Data (Test) =====**
TP = 28, FP = 2, TN = 475, FN = 55
Predictly Correct = 503
Predictly Wrong = 57

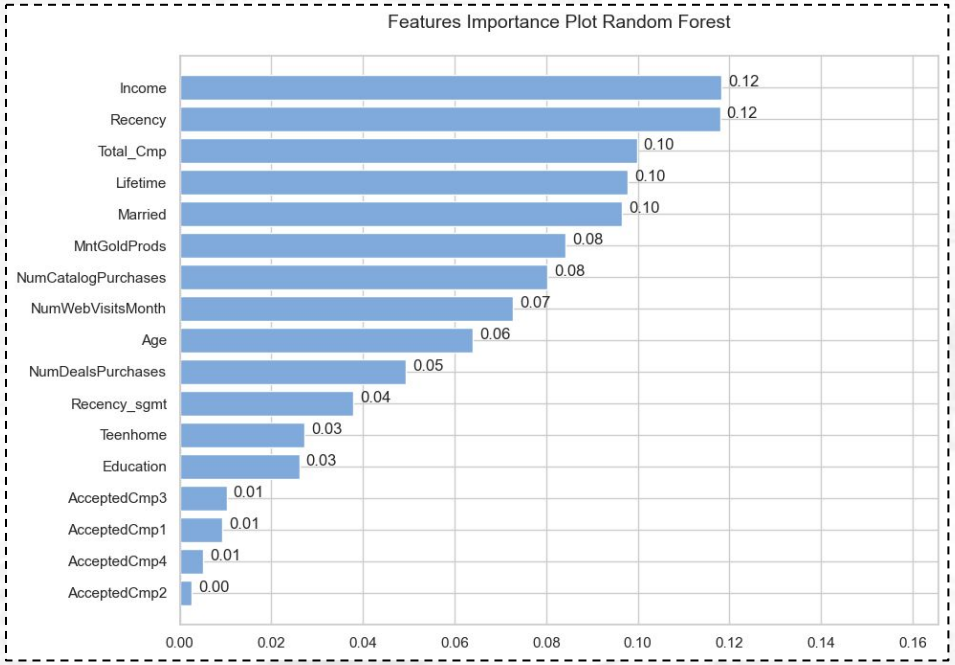
**Training Accuracy: 99.67 %**
**Test Accuracy: 90.89 %**



# Model Selection

## Feature Importance



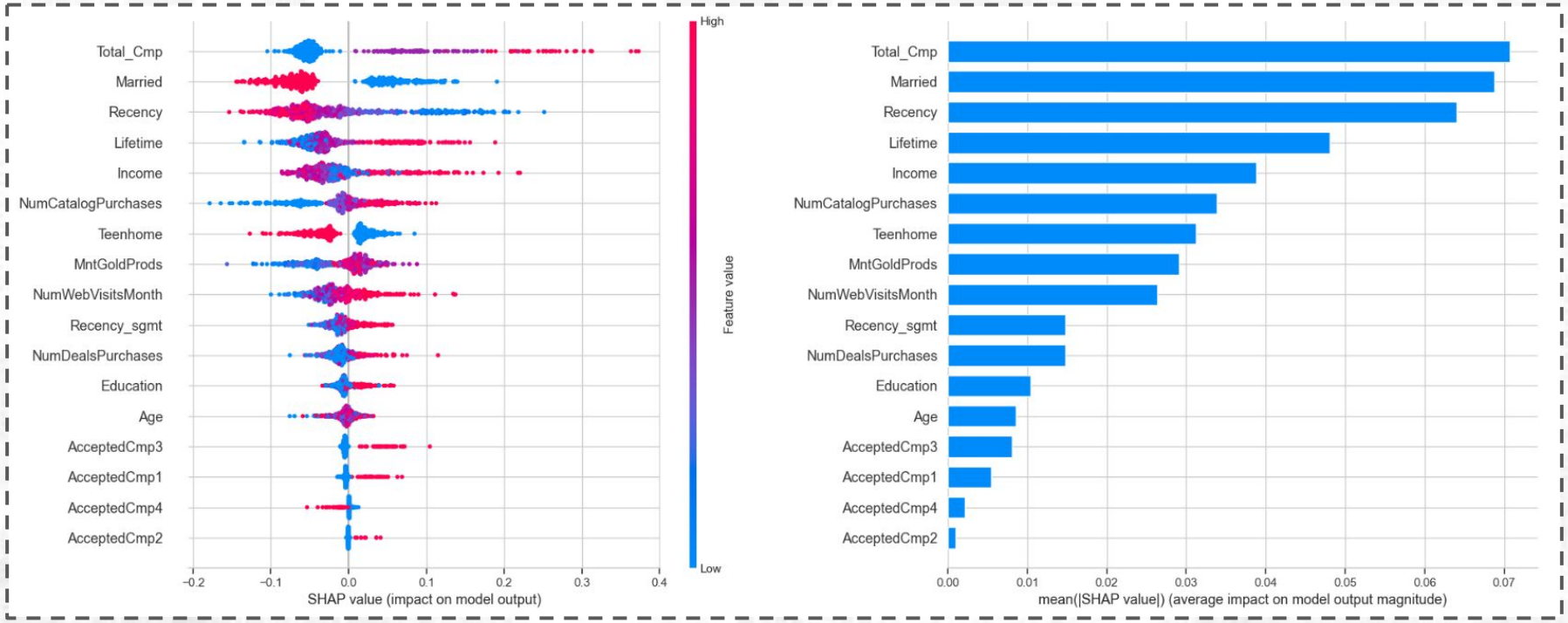

## Observation

Berikut adalah Top 10 variabel/feature yang paling berpengaruh terhadap target (Response)

- **Income** (pendapatan rumah tangga tahunan pelanggan) dengan feature importance score sekitar 0.118
- Recency (total hari terakhir customer berbelanja) dengan feature importance score sekitar 0.118
- **Total_camp** (total campaign yang pernah diterima/response oleh customer), dengan feature importance score paling tinggi sekitar 0.100
- **Lifetime** (total bulan customer berbelanja sejak pembelian pertama) dengan feature importance score 0.098
- **Married** (status kawin customer) dengan feature importance score sekitar 0.097
- **MntGoldProds** (total purchase Gold Products by customer) dengan feature importance score 0.084
- **NumCatalogPurchases** (pembelian produk melalui katalog atau printed list) dengan feature importance score sekitar 0.080
- **NumWebVisitMonth** (total pembelian produk melalui Web Visit bulan terakhir) dengan feature importance score 0.073
- **Age** (usia customer) dengan feature importance score sekitar 0.064
- **NumDealsPurchases** (total pembelian produk melalui diskon) dengan feature importance score 0.049

# Model Selection

## Shap Observation




# Model Selection

## Shap Observation

**Observation**

Berdasarkan SHAP diatas, dapat ditemukan beberapa insight sebagai berikut:

- **Total_camp** (total campaign yang pernah diterima/response oleh customer), feature valuenya cenderung berwarna merah yang mengarah kekanan (positif) menunjukkan bahwa semakin besar nilainya maka semakin besar pula  impact/pengaruh nilai untuk menerima campaign (response)
  - ➢ **Semakin tinggi total campaign yang diterima customer maka sangat besar peluang customer menerima campaign (response)**
- **Married** (customer yang berstatus belum/sudah menikah), feature valuenya cenderung berwarna biru yang mengarah kekanan (positif) menunjukkan bahwa semakin kecil nilainya atau jika customer belum menikah maka semakin besar pula impact/pengaruh nilai untuk menerima campaign (response)
  - ➢ **Kemungkinan peluang rate customer meresponse adalah pada status yang masih single**
- **Recency** (total hari terakhir customer berbelanja), feature valuenya cenderung berwarna biru yang mengarah kekanan (positif) menunjukkan bahwa semakin kecil nilainya atau semakin sering customer bebrbelanja maka semakin besar pula impact/pengaruh nilai untuk menerima campaign (response)
  - ➢ **Lama terakhir kali customer membeli products (recency) yang mana semakin kecil rentang waktunya akan sangat berpotensi untuk merespon campaign**
- **Lifetime** (total bulan customer berbelanja sejak pembelian pertama), feature valuenya cenderung berwarna merah yang mengarah kekanan (positif) menunjukkan bahwa semakin besar nilainya atau semakin sering/banyak total bulan customer untuk berbelaja maka semakin besar pula  impact/pengaruh nilai untuk menerima campaign (response)
  - ➢ **Lama customer berbelanja sejak hari pertama (lifetime/lama berlanggan) yang semakin besar rentang waktunya, maka akan berpeluang sangat besar untuk menerima campaign (customer langganan)**

# Model Selection

## Shap Observation (continue....)

**Observation**

Berdasarkan SHAP diatas, dapat ditemukan beberapa insight sebagai berikut:

- **Income** (pendapatan rumah tangga tahunan pelanggan), feature valuenya cenderung berwarna merah yang mengarah kekanan (positif), dan juga dipaling kiri menunjukkan bahwa semakin besar nilainya atau semakin sering/banyak pendapatan rumah tangga maka semakin besar pula impact/pengaruh nilai untuk menerima campaign (response), Namun ada juga Di area semakin sedikit pendapatan rumah tangga, juga berpotensi besar untuk menerima campaign (response)
  - ➢ **Semakin tinggi pendapatan rumah tangga tahunan pelanggan maka akan sangat berpotensi merespon campaign**
  - ➢ **Pada Middle Income cenderung tidak menerima campaign**
  - ➢ **Di area semakin sedikit pendapatan rumah tangga, juga berpotensi besar untuk menerima campaign (response)**
- **NumCatalogPurchases** (pembelian produk melalui katalog), feature valuenya cenderung berwarna merah yang mengarah kekanan (positif) menunjukkan bahwa semakin besar nilainya maka semakin besar pula impact/pengaruh nilai untuk menerima campaign (response)
  - ➢ **Customer yang mengunjungi toko offline lebih banyak cenderung berpeluang untuk merespons.**

**Observation** Dari Force Plots diatas, menunjukkan bahwa index data testing yang pertama yang tidak meresponse campaign dengan informasi yang didapat berdasarkan data tersebut menunjukkan bahwa:

- **Customer tidak pernah menerima campaign sebelumnya terbukti dari feature Total_Cmp = 0**
- **Customer sudah menikah terbukti dari feature Married = 1**
- **Customer tidak sering berbelanja terbukti dari feature Recency = 0.1717, Segment = 4**

# Business Insight and Recommendation

## 📌Conclusion

Berdasarkan modelling machine learning menggunakan metode **Random Forest** dengan hasil sebagai berikut :

**- Precision = 0.786**
**- Recall = 0.53**
**- F1 Score = 0.633**
**- ROC AUC = 0.905**

Didapatkan **Top 10 feature** yang paling berpengaruh untuk memprediksi model terhadap target/label Response diantaranya adalah:

- **Income** (pendapatan rumah tangga tahunan pelanggan)
- **Recency** (total hari terakhir customer berbelanja)
- **Total_camp** (total campaign yang pernah diterima/response oleh customer)
- **Lifetime** (total bulan customer berbelanja sejak pembelian pertama)
- **Married** (status kawin customer)
- **MntGoldProds** (total purchase Gold Products by customer)
- **NumCatalogPurchases** (pembelian produk melalui katalog atau printed list)
- **NumWebVisitMonth** (total pembelian produk melalui Web Visit bulan terakhir)
- **Age** (usia customer)
- **NumDealsPurchases** (total pembelian produk melalui diskon)

# Business Insight and Recommendation

## 📌Business Recommendation

**Actions**

- Memilih target marketing campaign pada customer yang memiliki Income (pendapatan) diatas ~75.000
- Melakukan penyaringan customer target dari marketing campaign pada Recency 0-20 hari
- Memberikan broadcast campaign kepada customer dengan total_cmp pada jumlah >=2
- Memberikan discount vouchers melalui Broadcast Message untuk customer yang memilki history response rate <2 atau <3 dari total 5 campaign yang pernah dilakukan
- Memperhatikan peningkatan broadcast campaign untuk total bulan customer berbelanja sejak pembelian pertama (Lifetime) pada sekitaran 32-35 bulan
- Jumlah customer paling banyak menerima respon berdasarkan status penikahan yaitu Married, sehingga pada campaign selanjutnya perusahaan sebaiknya memfokuskan kepada customer yang telah menikah (Married)
- Namun selain itu, untuk rate responce tertinggi sebesar 22.7% dari orang yang masih single, sehingga punya kesempatan kemungkinan besar menerima response lebih tinggi.
- Dapat memarakkan promosi product Gold pada karakteristik customer yang sesuai, dengan memberikan penawaran yang menarik melalui voucher diskon khusus atau paket promo.
- Pelanggan yang sering mengunjungi toko offline (Catalog Purchase) dan Yang mengunjungi website (Web Visit Month) memiliki peluang tinggi untuk merespon campaign. Perlu mempertahankan serta meningkatkan layanan dan fasilitas pada offline store yang ada. Selain itu meningkatkan engagement campaign pada website store agar memiliki pengalaman beli yang lancar, mudah dan nyaman.
- Kelompok umur yang paling banyak merespon campaign adalah Adult dan Senior Adult dan yang paling rendah adalah Young Adult. Artinya semakin Tua seseorang maka jumlah response juga meningkat. Kategori customer yang menerima Response terbanyak berasal dari tahun lahir 1970-1975 (39-44 years old), dan 1980-1990 (24-34 years old). Jika perusahaan harus memprioritaskan beberapa customer saja, maka perusahaan dapat memilih customer yang lahir pada tahun tersebut untuk menawarkan sebuah campaign.

# Business Insight and Recommendation

## 📌Business Recommendation

**Area Improvements**

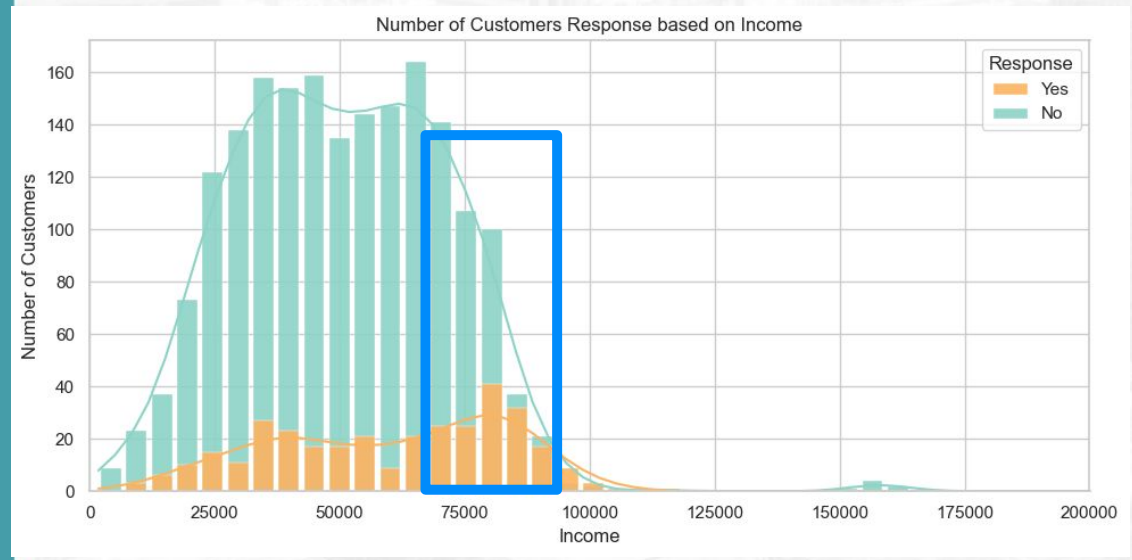
- Meningkatkan Response Rate dan Revenue serta mengurangi Cost biaya campaign dengan menentukan target market melalui Customer Segmentation (Cluster) yang tepat dilihat dari key features yang sudah ditentukan seperti Income, Recency, Total Campaign yang di terima oleh customer, Lifetime dan sebagainya.
- Mengadakan program loyalitas untuk mempertahankan pelanggan

**Reference**


- https://www.sciencedirect.com/science/article/pii/S2090447923001430
- https://www.sciencedirect.com/science/article/abs/pii/S1567422321000302
- https://www.sciencedirect.com/science/article/pii/S2667305323000601

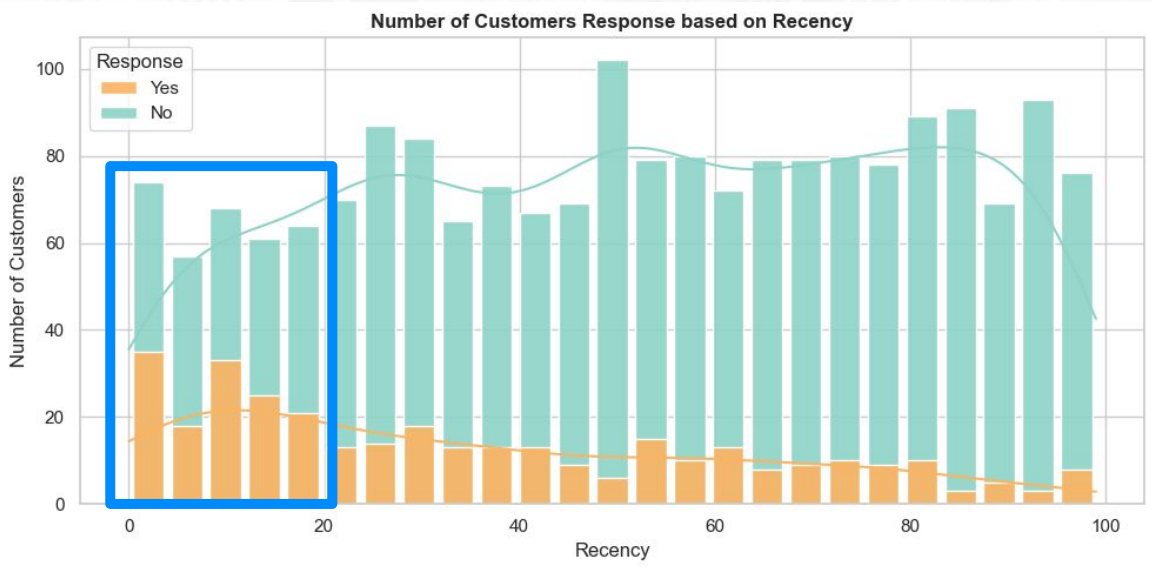
# Business Insight and Recommendation

**income >= $75K**


From the Income visualization, it can be seen that the customers who respond the most come from customers with income > $75000.

**Recency = 0-20 days**



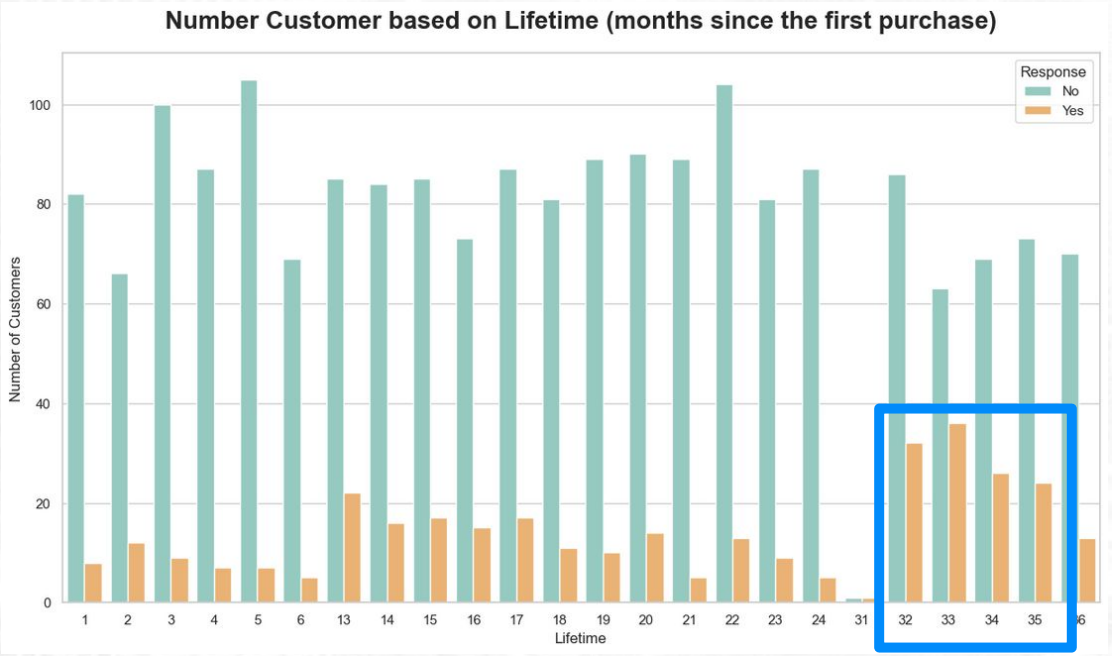

From the Recency visualization, it can be seen that customers who respond the most come from customers with low Recency.

# Business Insight and Recommendation

**Total Accepted Campaign by Customer >=2 (from 5 campaigns)**


**The most in our Five Campaigns is 0 (never responded), but there is a little potential in, only once (1) or twice (2) responded 325 and 83 Customers respectively.**

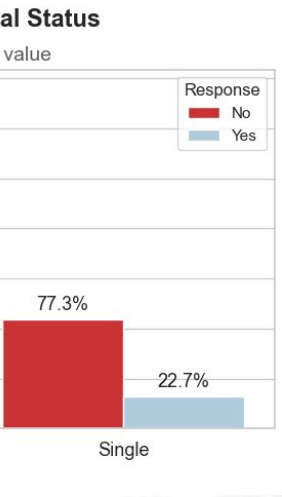
**Lifetime minimal 32 - 35 months**



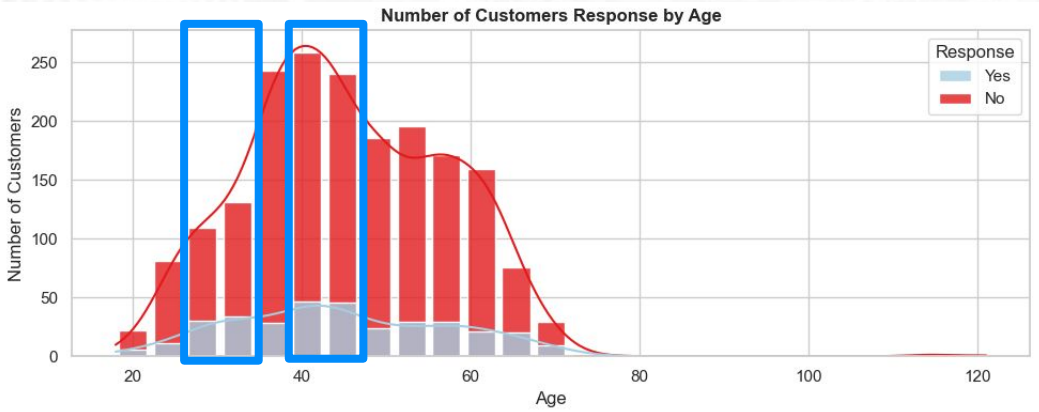

**Customers who have a high lifetime tend to respond to the campaign**

# Business Insight and Recommendation

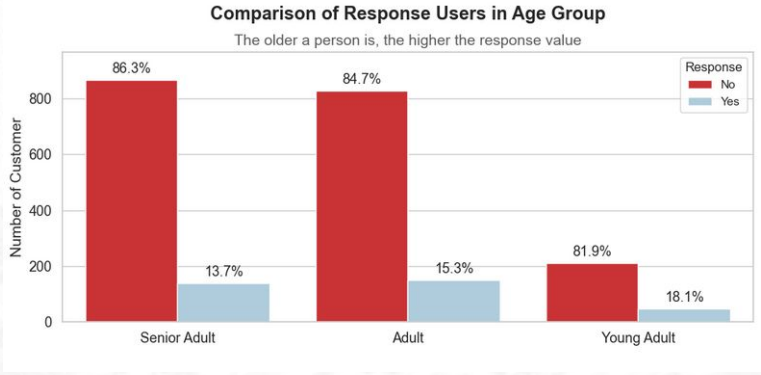
**Marrital Status =**
**Married > Single > Divorced**




**Age &**
**Age Group (dominated by Senior Adult group)**

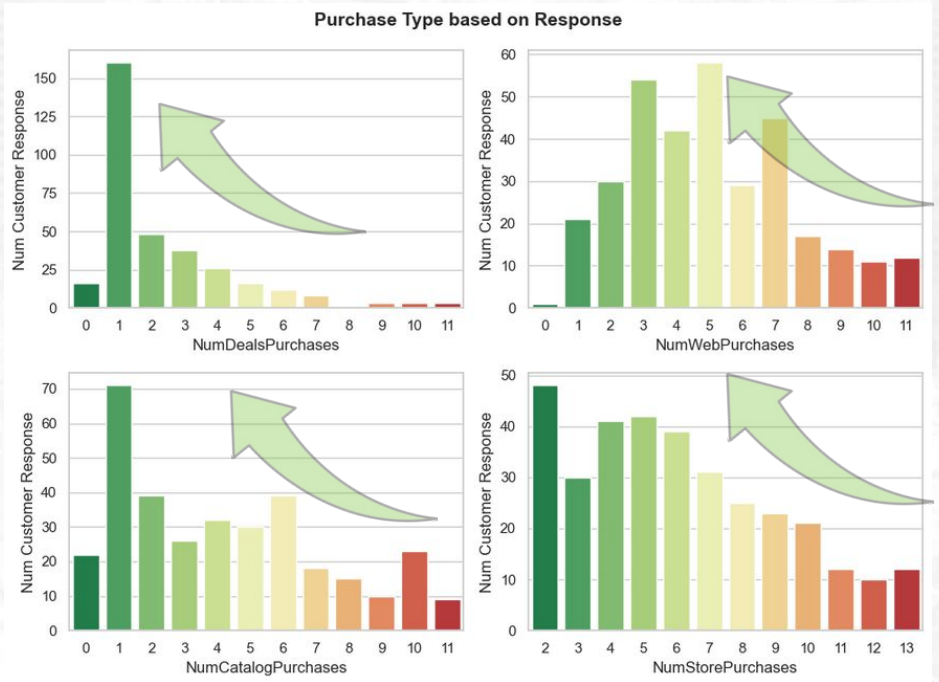



**More response in Single**
**The older a person is, the higher the response value**

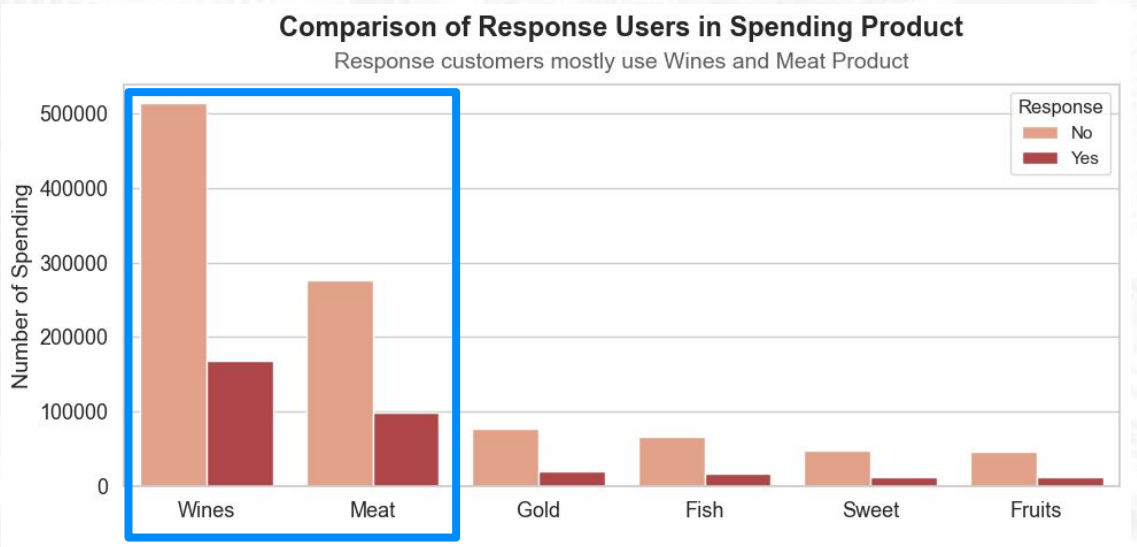


# Business Insight and Recommendation

**Responded Customer Purchase Types (the fewer purchases, the more likely it is)**




**Types of Purchased Products (customers tend to buy Wines and Meat)**



# Business Insight and Recommendation


## Discount/Flash Sale

price reduction with a certain period of time

especially for customers who have never response a campaign or 1x response the campaign only

Goals:
to increase customers shopping/purchasing interest


## Vouchers/Rewards

especially for customers who have response the campaign at least 2 campaigns

customers will get coupons and can be exchanged for discounts or other benefits such as points, free shipping, free gifts, etc.

Goals:
maintain customer's interest and retention


## Promo Bundling/Special Offer

can implement for all customers

combining the main product with supporting products for more economical price for customers such as Wines with Meats, Wines with Gold Products, etc.

Goals:
encourage customers to buy more products

# Business Simulation

## Response Rate

**14,91%**

before model

**response rate = TP/(TP+FP)**

**78,57%**

after model

**63,66**

**True Positive (TP)= 44**

**False Positive (FP)= 12**

**Sebelum Modelling**

```
a = df_all['Response'].value_counts()/len(df_all['Response'])
response_ratio =  pd.DataFrame(a).T
response_ratio = response_ratio.rename(columns={0:'No Response' , 1:'Response'
response_ratio = response_ratio*100
print("Response / Acceptance Rate (Before Modelling) =", round(response_ratio[
```

```
Response / Acceptance Rate (Before Modelling) = 14.91 %
```

**Setelah Modelling**

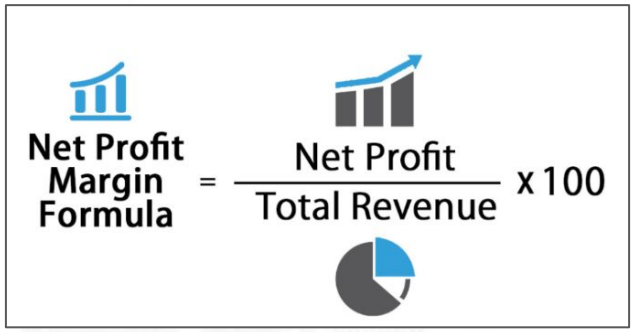
```
# response_rate is derived from predicted data = 1 (True)
true_positive = 44
false_positive = 12
# Precision
response_rate = (true_positive/(true_positive+false_positive)) * 100
print("Response / Acceptance Rate (After Modelling) =", round(response_rate, 2
```

```
Response / Acceptance Rate (After Modelling) = 78.57 %
```

Dari hasil perhitungan, terbukti bahwa dengan menerapkan machine learning melalui teknik modelling Random Forest, pada kasus ini, perusahaan supermarket lotte mart dapat berpotensi meningkatkan response rate dari yang sebelumnya **hanya sebesar 14.9 % menjadi 78.5 %.**

# Business Simulation

## Net Profit Margin / NPM




### Before Modeling

```
# basis
revenue = 11
cost = 3

# total campaign predict positif
total_campaign = tp + fp

print("Total Response = {}".format(round(tp, 2)))
print("Total Campaign = {}".format(round(total_campaign, 2)))

# calculate cost revenue profit
total_cost = total_campaign * cost
total_revenue = tp * revenue
total_profit = total_revenue - total_cost

print("Total Cost = ${}".format(round(total_cost, 2)))
print("Total Revenue = ${}".format(round(total_revenue, 2)))
print("Total Profit = ${}".format(round(total_profit, 2)))

# calculate rate
revenue_rate_after = (total_profit/ total_revenue) *100
print("------------------------------")
print("Revenue/Profit Rate (After Modelling) =", round(revenue_rate_after, 2),
```

```
====== After Modelling ======
Total Response = 44
Total Campaign = 56
Total Cost = $168
Total Revenue = $484
Total Profit = $316
------------------------------
Revenue/Profit Rate (After Modelling) = 65.29 %
```

### After Modeling

```
# basis
revenue = 11
cost = 3

# total campaign for all customers
total_campaign = len(df_all)

print("Total Response = {}".format(round(total_respon, 2)))
print("Total Campaign = {}".format(round(total_campaign, 2)))

# calculate cost revenue profit
total_cost = total_campaign * cost
total_revenue = total_respon * revenue
total_profit = total_revenue - total_cost

print("Total Cost = ${}".format(round(total_cost, 2)))
print("Total Revenue = ${}".format(round(total_revenue, 2)))
print("Total Profit = ${}".format(round(total_profit, 2)))

# calculate rate
revenue_rate_before = (total_profit/ total_revenue) *100
print("------------------------------")
print("Revenue/Profit Rate (Before Modelling) =", round(revenue_rate_before, 2)
```

```
====== Before Modelling ======
Total Response = 334
Total Campaign = 2240
Total Cost = $6720
Total Revenue = $3674
Total Profit = $-3046
------------------------------
Revenue/Profit Rate (Before Modelling) = -82.91 %
```

# Business Simulation

## Net Profit Margin / NPM

### Before Modeling

**-82,91%**

| | | |
|---|---|---|
| **Total Response (1=accept response)** | | **334** |
| **Total Campaign (all customers)** | | **2240** |
| **Total Cost($)** | = Total campaign*cost<br>= 2240 * 3 | **6720** |
| **Total Revenue($)** | = Total response * revenue<br>= 334 * 11 | **3674** |
| **Total Profit($)** | = Total Revenue - Total Cost<br>= 3674 - 6720 | **-3046** |
| **NPM (%)** | = (Total Profit/ Total Revenue) *100 | **-82.91** |

**148,2%**

a customer

| | |
|---|---|
| **Revenue ($)** | **11** |
| **Cost ($)** | **3** |

**Broadcast Message**
- SMS/WA blast
- paid vendor
- segmented customer

**Digital Ads**
- paid ads
- ads in Google, social media

### After Modeling

**65,29%**

| | | |
|---|---|---|
| **Total Campaign (TP+FP)** | | **56** |
| **Total Cost($)** | = Total campaign*cost<br>= 56 * 3 | **168** |
| **Total Revenue($)** | = TP * revenue<br>= 44 * 11 | **484** |
| **Total Profit($)** | = Total Revenue-Total Cost<br>= 484-168 | **316** |
| **NPM (%)** | = (Total Profit/ Total Revenue) *100 | **65.29** |

# Business Simulation

## Return of Investment

**ROI adalah singkatan dari Return On Investment** yang merupakan rasio untuk melakukan perhitungan efektivitas sebuah investasi yang diberikan. Secara teknis, ROI adalah perhitungan laba bersih yang kita dapatkan dari nominal uang investasi yang sudah dikeluarkan.

1. **Efisiensi Penggunaan Dana**
   Fungsi pertama dari ROI adalah saat sebuah perusahaan sudah menjalankan praktek akuntansi yang baik maka manajemen dengan menggunakan teknik analisa ROI dapat mengukur efisiensi penggunaan modal
2. **Mengetahui Kelemahan Perusahaan**
   Manfaat berikutnya dari ROI adalah dapat membandingkan efisiensi penggunaan modal pada perusahaannya dengan perusahaan lain yang sejenis, sehingga dapat diketahui apakah perusahaannya berada di bawah, sama, atau diatas rata-ratanya.
3. **Mengukur Profitabilitas sebuah Perusahaan/Produk**
   Jika sebuah perusahaan menggunakan “product cost system” yang baik, dari perhitungan modal dan biaya dapat dialokasikan kepada berbagai-bagai produk yang dihasilkan oleh perusahaan
4. **Sebagai Alat Kontrol**
   Fungsi terakhir dari ROI adalah sebagai alat kontrol untuk melihat bagaimana prospek saat ini sehingga bisa mendapatkan bayangan langkah apa yang selanjutnya bisa diambil oleh perusahaan.

```
====== CLV ======
Total Customer / Order = 2240
Total Respon = 334
Total Revenue = Total Respon * 11 = 24640
--------------------------------------------------
Average Order Size (AOS) = Total Revenue / Total Customer = 11.0
Average Order Frequency (AOF) = Total Order / Total Customer = 1.0
Average Customer Value (ACV) = AOS / AOF = 11.0
Average Customer Lifetime (ACL) = 1 Year
Customer Lifetime Value (CLV) = 11.0
```

### Sebelum Modelling

```
total_cost = total_customer * cost
cac = total_cost/total_respon
roi = clv/cac
print("Return of Investment (Before Modelling) =", round(roi, 2))
```

```
Return of Investment (Before Modelling) = 0.55
```

### Setelah Modelling

```
total_cost_am = (tp+fp) * cost
cac = total_cost_am/tp
roi = clv/cac
print("Return of Investment (After Modelling) =", round(roi, 2))
```

```
Return of Investment (After Modelling) = 2.88
```

# Business Simulation

## Return of Investment

| | | |
|---|---|---|
| Average Order Size (AOS) | = Total Revenue/Total Order<br>= (2240*11)/2240<br>= 24640/2240 | 11 |
| Average Order Frequency (AOF) | = Total Order/Total Customer<br>= 2240/2240 | 1 |
| Average Customer Value (ACV) | = AOS/AOF | 11 |
| Average Cust. Llifetime (ACL)(Year) | = first order date-last order date | 1 |
| Customer Lifetime Value (CLV) ($) | = ACL*ACV | 11 |

ROI = ~3.0x

- **Good ROI (Manzer 2017)**
- **The company receives $3 per $1 spent to acquire the customer**

| Before Modelling | | |
|---|---|---|
| number of new customers | | 334 |
| CLV | | 11 |
| CAC | = Total Marketing Cost/Number new Customers<br>= (2240*3)/ 334 | 20.1 |
| ROI | = CLV : CAC | 0.5 |

| After Modelling | | |
|---|---|---|
| number of new customers | | 44 |
| CLV | | 11 |
| CAC | = Total Marketing Cost/Number new Customers<br>= ((TP+FP)*cost) / 44<br>= (56*3) / 44 | 3.8 |
| ROI | = CLV : CAC | 2.8 |

Rakamin Academy

# Clustering Models

# Feature Scaling

untuk membuat numerical data pada dataset memiliki rentang nilai (scale) yang sama maka diperlukan feature scaling

```
# for standardization
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(clus_df)
clus_dfsc = pd.DataFrame(scaler.transform(clus_df), columns= clus_df.columns )
clus_dfsc.describe()
```

| | Education | Income | Kidhome | Teenhome | Recency | MntWines | MntFruits | MntMeatProducts | MntFishProducts | MntSweetProducts |
|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 2236.000000 | 2236.000000 | 2236.000000 | 2236.000000 | 2236.000000 | 2236.000000 | 2236.000000 | 2236.000000 | 2236.000000 | 2236.000000 |
| **mean** | -0.000000 | -0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | -0.000000 |
| **std** | 1.000224 | 1.000224 | 1.000224 | 1.000224 | 1.000224 | 1.000224 | 1.000224 | 1.000224 | 1.000224 | 1.000224 |
| **min** | -1.983191 | -2.335766 | -0.824939 | -0.930615 | -1.696543 | -1.970872 | -1.454849 | -2.511714 | -1.509682 | -1.434003 |
| **25%** | -0.792105 | -0.773713 | -0.824939 | -0.930615 | -0.867550 | -0.910486 | -0.985311 | -0.838594 | -0.700799 | -0.971157 |
| **50%** | -0.792105 | -0.028350 | -0.824939 | -0.930615 | -0.004016 | 0.126169 | -0.001432 | 0.034712 | 0.003254 | -0.001681 |
| **75%** | 0.398982 | 0.766366 | 1.032627 | 0.905974 | 0.859517 | 0.874655 | 0.829614 | 0.842366 | 0.838983 | 0.816780 |
| **max** | 1.590069 | 5.132368 | 2.890194 | 2.742564 | 1.723051 | 1.812958 | 1.884177 | 2.222716 | 1.862577 | 2.012128 |

# Identyfing the Numbers of Clusters

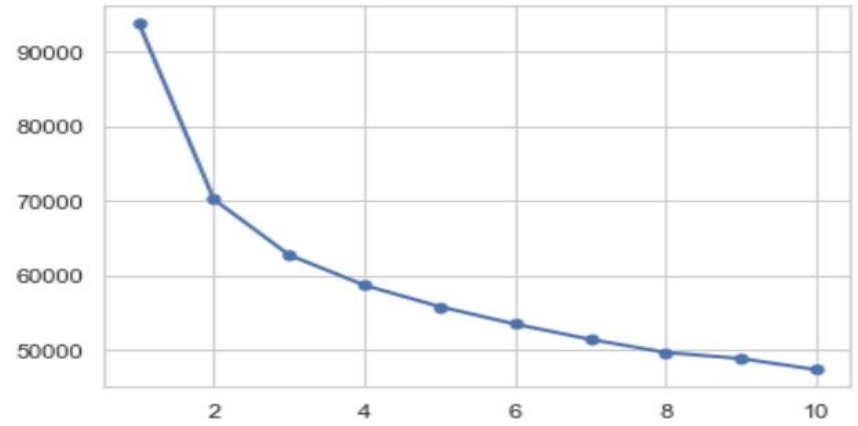
Untuk menentukan jumlah cluster yang optimal, kita harus memilih nilai k pada "siku", yaitu titik di mana distorsi/inersia mulai menurun secara linier.

**Elbow Method**

```
from sklearn.cluster import KMeans
inertia = []
for i in range(1, 11):
    km = KMeans(n_clusters=i, init='k-means++', random_state=0)
    km.fit(clus_dfsc)
    inertia.append(km.inertia_)
```

```
sns.lineplot(x=range(1, 11), y=inertia, linewidth=2)
sns.scatterplot(x=range(1, 11), y=inertia, s=50)
```

```
<AxesSubplot:>
```




```
(pd.Series(inertia) - pd.Series(inertia).shift(-1)) / pd.Series(inertia) * 100
```

```
0   25.252919
1   10.632353
2    6.473955
3    4.885154
4    4.189489
5    3.849898
6    3.403908
7    1.657230
8    3.037698
9         NaN
dtype: float64
```

Berdasarkan hasil penentuan jumlah clustering dengan elbow method perubahan tidak berubah signifikan pada 4 cluster, maka akan dibagi menjadi ke 4 cluster

# KMeans Clustering Without Dimensionality Reduction

```
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0)
kmeans.fit(clus_dfsc)
#Adding the Clusters feature to the orignal dataframe.
clus_df["cluster_km_orig"] = kmeans.labels_
df["cluster_KM_orig"] = kmeans.labels_
# Get the predictions
predictions = kmeans.predict(clus_dfsc)
clus_df.head()
```

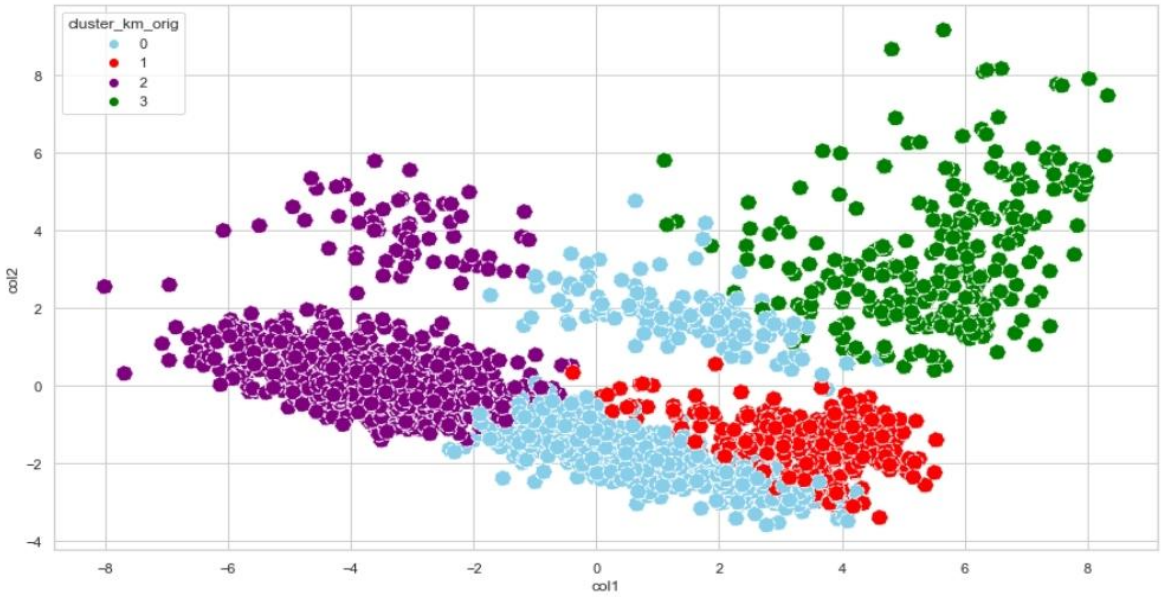
| | Education | Income | Kidhome | Teenhome | Recency | MntWines | MntFruits |
|---|---|---|---|---|---|---|---|
| **0** | 1 | 0.351086 | 0 | 0 | 0.585859 | 1.058011 | 1.409686 |
| **1** | 1 | 0.277680 | 1 | 1 | 0.383838 | -1.212159 | -0.985311 |
| **2** | 1 | 0.434956 | 0 | 0 | 0.262626 | 0.745578 | 1.064234 |
| **3** | 1 | 0.155079 | 1 | 0 | 0.262626 | -1.212159 | -0.380323 |
| **4** | 3 | 0.352051 | 1 | 0 | 0.949495 | 0.122520 | 0.986785 |

Menambahkan clusters feature ke dataframe asli

Hasil visualisasi

```
PCA_ds["cluster_km_orig"] = clus_df["cluster_km_orig"].copy()
# Visualize the results in 2D
fig, ax = plt.subplots(figsize=(15,8))

sns.scatterplot(
    x="col1", y="col2",
    hue="cluster_km_orig",
    linestyle='--',
    data=PCA_ds,
    palette=['skyblue','red','purple','green'],
    s=160,
    ax=ax
)
```

```
<AxesSubplot:xlabel='col1', ylabel='col2'>
```



# KMeans Clustering With Dimensionality Reduction

```
PCA_ds.drop("cluster_km_orig", axis=1, inplace=True)
PCA_ds.head()
```

Kolom "cluster_km_orig" telah dihapus dari dataframe PCA_ds. Kemudian, perintah PCA_ds.head() digunakan untuk melihat hasilnya, yaitu sebagian data teratas dari dataframe setelah kolom tersebut dihapus.

**Hasil :**

| | col1 | col2 |
|---|---|---|
| 0 | 4.170015 | -1.607306 |
| 1 | -4.347667 | 0.065993 |
| 2 | 3.331285 | -1.537009 |
| 3 | -3.964544 | 0.668766 |
| 4 | 0.608980 | -1.441096 |

Setiap baris mewakili sepasang nilai dalam kolom "col1" dan "col2". Data ini dapat digunakan sebagai input untuk berbagai analisis atau visualisasi yang melibatkan dua variabel.

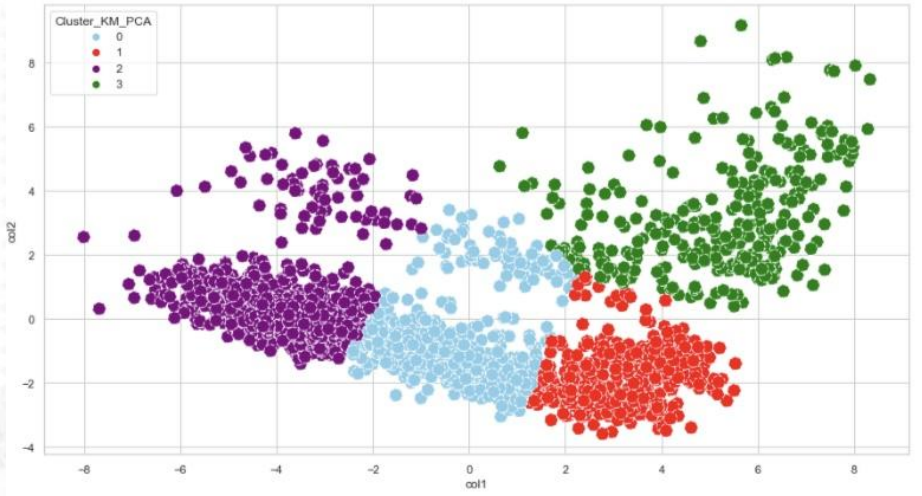
```
#Use PCA for Cluster
kMeans = KMeans(n_clusters = 4, init = 'k-means++', random_state=0)
y_pred_kMeans = kMeans.fit_predict(PCA_ds)
PCA_ds["Cluster_KM_PCA"] = y_pred_kMeans
#Adding the Clusters feature to the orignal dataframe.
clus_df["Cluster_KM_PCA"] = y_pred_kMeans
df["cluster_KM_PCA"] = y_pred_kMeans
```

K-Means dilatih pada data PCA_ds, dan label kluster yang dihasilkan kemudian ditambahkan ke dataframe PCA_ds, clus_df, dan df untuk keperluan analisis selanjutnya

**Melakukan Visualisasi Hasil**

```
# Visualize the results in 2D
fig, ax = plt.subplots(figsize=(15,8))

sns.scatterplot(
    x="col1", y="col2",
    hue="Cluster_KM_PCA",
    linestyle='--',
    data=PCA_ds,
    palette=['skyblue','red','purple','green'],
    s=160,
    ax=ax
)
```

**Hasil :**



# Clustering With RFM Analysis

```
df_rfm['Spending']=df_rfm['MntWines']+ \
    df_rfm['MntFruits']+ \
    df_rfm['MntMeatProducts']+ \
    df_rfm['MntFishProducts']+ \
    df_rfm['MntSweetProducts']+ \
    df_rfm['MntGoldProds']
```

```
df_rfm['Total_Purchases'] = df_rfm['NumDealsPurchases'] + \
    df_rfm['NumWebPurchases'] + \
    df_rfm['NumCatalogPurchases'] + \
    df_rfm['NumStorePurchases']
```

```
df_rfm=df_rfm[['ID','Spending','Total_Purchases','Recency','Dt_Customer']]
df_rfm.head()
```

| | ID | Spending | Total_Purchases | Recency | Dt_Customer |
|---|---|---|---|---|---|
| 0 | 5524 | 1617 | 25 | 58 | 2012-09-04 |
| 1 | 2174 | 27 | 6 | 38 | 2014-03-08 |
| 2 | 4141 | 776 | 21 | 26 | 2013-08-21 |
| 3 | 6182 | 53 | 8 | 26 | 2014-02-10 |
| 4 | 5324 | 422 | 19 | 94 | 2014-01-19 |

Syntax diatas menghasilkan DataFrame baru yang berisi kolom-kolom "ID", "Spending", "Total_Purchases", "Recency", dan "Dt_Customer" dari DataFrame asli df_rfm. Kolom "Spending" menggabungkan beberapa kolom lain untuk menghitung total pengeluaran pelanggan, sedangkan kolom "Total_Purchases" menggabungkan beberapa kolom lain untuk menghitung total pembelian pelanggan.

## Melakukan Recency, Frequency, Monetary Calculation, dab DataFrame Aggregation

```
recency_df = df_rfm[['ID','Recency']]
recency_df.head()
```

| | ID | Recency |
|---|---|---|
| 0 | 5524 | 58 |
| 1 | 2174 | 38 |
| 2 | 4141 | 26 |
| 3 | 6182 | 26 |
| 4 | 5324 | 94 |

```
frequency_df = df_rfm[['ID','Total_Purchases']]
temp_df = recency_df.merge(frequency_df,on='ID')
frequency_df.head()
```

| | ID | Total_Purchases |
|---|---|---|
| 0 | 5524 | 25 |
| 1 | 2174 | 6 |
| 2 | 4141 | 21 |
| 3 | 6182 | 8 |
| 4 | 5324 | 19 |

```
monetary_df = df_rfm[['ID','Spending']]
monetary_df.head()
```

| | ID | Spending |
|---|---|---|
| 0 | 5524 | 1617 |
| 1 | 2174 | 27 |
| 2 | 4141 | 776 |
| 3 | 6182 | 53 |
| 4 | 5324 | 422 |

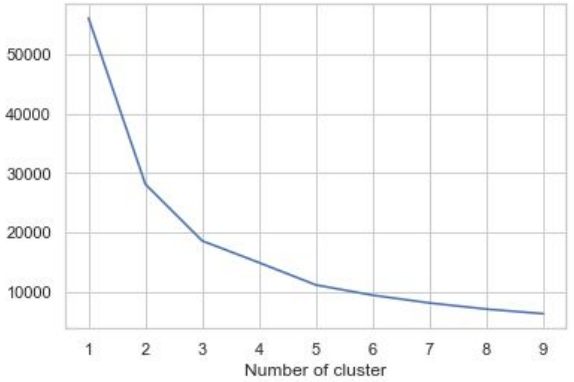
```
tx_user  = temp_df.merge(monetary_df,on='ID')
tx_user.columns = ['ID','Recency','Frequency','Monetary']
tx_user.head()
```

| | ID | Recency | Frequency | Monetary |
|---|---|---|---|---|
| 0 | 5524 | 58 | 25 | 1617 |
| 1 | 2174 | 38 | 6 | 27 |
| 2 | 4141 | 26 | 21 | 776 |
| 3 | 6182 | 26 | 8 | 53 |
| 4 | 5324 | 94 | 19 | 422 |

## Elbow Method

```
from sklearn_extra.cluster import KMedoids
sse={}
tx_recency = tx_user[['Recency']]
for k in range(1, 10):
    kmedoids = KMedoids(n_clusters=k, random_state=0, max_iter=1000,init='k-medoids++',metric='euclidean').fit(tx_recency)
    tx_recency["clusters"] = kmedoids.labels_
    sse[k] = kmedoids.inertia_
plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster")
plt.show()
```

Elbow Method melakukan iterasi melalui berbagai jumlah klaster, melatih model K-Medoids pada fitur "Recency" dari data pelanggan, dan menghitung nilai SSE untuk setiap jumlah klaster. Kemudian, kode tersebut memplotkan grafik jumlah klaster versus SSE, yang dapat membantu dalam menentukan jumlah cluster yang optimal untuk analisis klasterisasi.



# Clustering With RFM Analysis

Dalam analisis ini, kami akan membagi pelanggan kami ke dalam 5 klaster untuk setiap metrik RFM, menghasilkan 5x5x5 klaster.

## Recency Cluster Creation

```
[ ] kmedoids = KMedoids(n_clusters=5, random_state=0, max_iter=1000,init='k-medoids++',metric='euclidean').fit(tx_recency)
    tx_user['RecencyCluster'] = kmedoids.predict(tx_recency)

    #function for ordering cluster numbers
    def order_cluster(cluster_field_name, target_field_name,df,ascending):
        new_cluster_field_name = 'new_' + cluster_field_name
        df_new = df.groupby(cluster_field_name)[target_field_name].mean().reset_index()
        df_new = df_new.sort_values(by=target_field_name,ascending=ascending).reset_index(drop=True)
        df_new['index'] = df_new.index
        df_final = pd.merge(df,df_new[[cluster_field_name,'index']], on=cluster_field_name)
        df_final = df_final.drop([cluster_field_name],axis=1)
        df_final = df_final.rename(columns={"index":cluster_field_name})
        return df_final

    tx_user = order_cluster('RecencyCluster', 'Recency',tx_user,False)
    #see details of each cluster
    tx_user.groupby('RecencyCluster')['Recency'].describe()
```

| RecencyCluster | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| 0 | 392.000000 | 90.418367 | 5.137841 | 82.000000 | 86.000000 | 91.000000 | 95.000000 | 99.000000 |
| 1 | 430.000000 | 72.153488 | 5.593009 | 63.000000 | 67.000000 | 72.000000 | 77.000000 | 81.000000 |
| 2 | 472.000000 | 52.027542 | 5.706301 | 42.000000 | 47.750000 | 52.000000 | 56.000000 | 62.000000 |
| 3 | 490.000000 | 30.240816 | 6.173296 | 20.000000 | 25.000000 | 30.000000 | 36.000000 | 41.000000 |
| 4 | 456.000000 | 9.122807 | 5.849227 | 0.000000 | 4.000000 | 9.000000 | 14.000000 | 19.000000 |

## Monetary Cluster Creation

```
tx_monetary = tx_user[['Monetary']]

kmedoids = KMedoids(n_clusters=5, random_state=0, max_iter=1000,init='k-medoids++',metric='euclidean').fit(tx_monetary)
tx_user['MonetaryCluster'] = kmedoids.predict(tx_monetary)

#order the cluster numbers
tx_user = order_cluster('MonetaryCluster', 'Monetary',tx_user,True)

#show details of the dataframe
tx_user.groupby('MonetaryCluster')['Monetary'].describe()
```

| MonetaryCluster | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| 0 | 939.000000 | 74.423855 | 53.566384 | 5.000000 | 36.000000 | 57.000000 | 97.500000 | 231.000000 |
| 1 | 398.000000 | 407.756281 | 110.683876 | 232.000000 | 311.000000 | 406.000000 | 493.750000 | 615.000000 |
| 2 | 314.000000 | 828.961783 | 113.620066 | 622.000000 | 731.000000 | 832.500000 | 928.000000 | 1009.000000 |
| 3 | 315.000000 | 1208.228571 | 122.867096 | 1012.000000 | 1102.500000 | 1189.000000 | 1315.000000 | 1445.000000 |
| 4 | 274.000000 | 1766.171533 | 243.027162 | 1449.000000 | 1574.000000 | 1701.500000 | 1919.000000 | 2525.000000 |

## Frequency Cluster Creation

```
tx_frequency = tx_user[['Frequency']]

kmedoids = KMedoids(n_clusters=5, random_state=0, max_iter=1000,init='k-medoids++',metric='euclidean').fit(tx_frequency)
tx_user['FrequencyCluster'] = kmedoids.predict(tx_frequency)

#order the frequency cluster
tx_user = order_cluster('FrequencyCluster', 'Frequency',tx_user,True)

#see details of each cluster
tx_user.groupby('FrequencyCluster')['Frequency'].describe()
```

| FrequencyCluster | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| 0 | 832.000000 | 6.608173 | 2.012077 | 0.000000 | 5.000000 | 7.000000 | 8.000000 | 10.000000 |
| 1 | 310.000000 | 12.996774 | 1.493239 | 11.000000 | 12.000000 | 13.000000 | 14.000000 | 15.000000 |
| 2 | 491.000000 | 17.940937 | 1.425201 | 16.000000 | 17.000000 | 18.000000 | 19.000000 | 20.000000 |
| 3 | 400.000000 | 22.770000 | 1.398925 | 21.000000 | 22.000000 | 23.000000 | 24.000000 | 25.000000 |
| 4 | 207.000000 | 28.246377 | 2.808132 | 26.000000 | 26.000000 | 27.000000 | 29.000000 | 44.000000 |

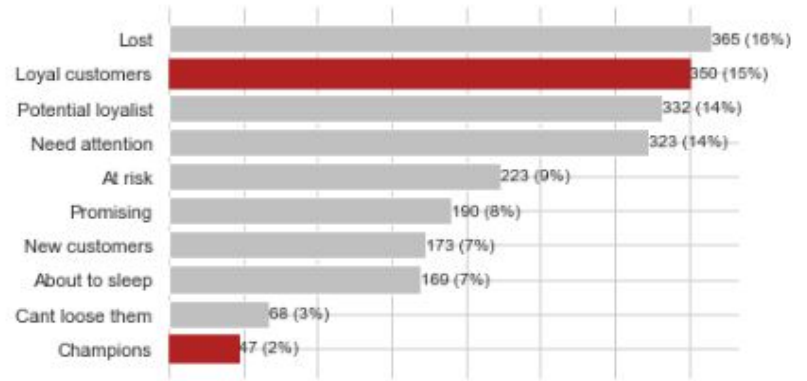
# Clustering With RFM Analysis

### Segment Creation

```
segt_map = {
    r'30': 'Promising',
    r'23': 'Loyal customers',
    r'24': 'Loyal customers',
    r'33': 'Loyal customers',
    r'34': 'Loyal customers',
    r'43': 'Loyal customers',
    r'32': 'Potential loyalist',
    r'31': 'Potential loyalist',
    r'42': 'Potential loyalist',
    r'41': 'Potential loyalist',
    r'21': 'Need attention',
    r'22': 'Need attention',
    r'12': 'Need attention',
    r'11': 'Need attention',
    r'40': 'New customers',
    r'20': 'About to sleep',
    r'14': 'Cant loose them',
    r'04': 'Cant loose them',
    r'10': 'Lost',
    r'00': 'Lost',
    r'01': 'Lost',
    r'02': 'At risk',
    r'03': 'At risk',
    r'13': 'At risk',
    r'44': 'Champions',
}

tx_user['Segment'] = tx_user['RecencyCluster'].map(str) + tx_user['FrequencyCluster'].map(str)
tx_user['Segment'] = tx_user['Segment'].replace(segt_map, regex=True)
tx_user.sample(10)
```

| | ID | Recency | Frequency | Monetary | RecencyCluster | FrequencyCluster | MonetaryCluster | Segment |
|---|---|---|---|---|---|---|---|---|
| **213** | 590 | 35 | 16 | 1812 | 3 | 2 | 4 | Potential loyalist |
| **561** | 8560 | 94 | 18 | 680 | 0 | 2 | 2 | At risk |
| **2092** | 3798 | 80 | 9 | 81 | 1 | 0 | 0 | Lost |
| **1856** | 10270 | 8 | 8 | 66 | 4 | 0 | 0 | New customers |
| **1636** | 255 | 31 | 6 | 21 | 3 | 0 | 0 | Promising |
| **9** | 3749 | 54 | 24 | 1580 | 2 | 3 | 4 | Loyal customers |
| **2167** | 9432 | 23 | 11 | 62 | 3 | 1 | 0 | Potential loyalist |
| **1579** | 2173 | 23 | 5 | 32 | 3 | 0 | 0 | Promising |
| **1885** | 3885 | 10 | 5 | 63 | 4 | 0 | 0 | New customers |
| **2020** | 5555 | 81 | 0 | 6 | 1 | 0 | 0 | Lost |

```
# count the number of customers in each segment
segments_counts = tx_user['Segment'].value_counts().sort_values(ascending=True)

fig, ax = plt.subplots()

bars = ax.barh(range(len(segments_counts)),
               segments_counts,
               color='silver')
ax.set_frame_on(False)
ax.tick_params(left=False,
               bottom=False,
               labelbottom=False)
ax.set_yticks(range(len(segments_counts)))
ax.set_yticklabels(segments_counts.index)

for i, bar in enumerate(bars):
        value = bar.get_width()
        if segments_counts.index[i] in ['Champions', 'Loyal customers']:
            bar.set_color('firebrick')
        ax.text(value,
                bar.get_y() + bar.get_height()/2,
                '{:,} ({:}%)'.format(int(value),
                                     int(value*100/segments_counts.sum())),
                va='center',
                ha='left'
               )

plt.show()
```



# Clustering With RFM Analysis

## Metric Analysis per segment

```
# Calculate average values for each RFM segment, and return a size of each segment
tx_user_viz = tx_user.groupby('Segment').agg({
    'Recency': 'mean',
    'Frequency': 'mean',
    'Monetary': ['mean', 'count'],
}).round(1)
# Print the aggregated dataset
tx_user_viz
```

| Segment | Recency mean | Frequency mean | Monetary mean | Monetary count |
|---|---|---|---|---|
| About to sleep | 51.500000 | 6.600000 | 72.100000 | 169 |
| At risk | 84.800000 | 21.100000 | 1077.500000 | 223 |
| Cant loose them | 81.200000 | 28.400000 | 1230.500000 | 68 |
| Champions | 9.200000 | 27.800000 | 1039.100000 | 47 |
| Lost | 82.800000 | 7.800000 | 134.200000 | 365 |
| Loyal customers | 35.300000 | 24.200000 | 1136.500000 | 350 |
| Need attention | 62.000000 | 16.000000 | 773.000000 | 323 |
| New customers | 9.500000 | 6.500000 | 64.400000 | 173 |
| Potential loyalist | 19.200000 | 16.100000 | 758.000000 | 332 |
| Promising | 30.600000 | 6.700000 | 67.200000 | 190 |

```
newr= tx_user_viz['Recency']['mean'].apply(lambda x: 'medium' if x>=np.quantile(tx_user_viz['Recency']['mean'], .25) and
                                           x<=np.quantile(tx_user_viz['Recency']['mean'], .75) else 'high' if x>np.quantile(tx_user_viz['Recency']['mean'], .75) else 'low')
newf= tx_user_viz['Frequency']['mean'].apply(lambda x:'medium' if x>=np.quantile(tx_user_viz['Frequency']['mean'], .25) and
                                             x<=np.quantile(tx_user_viz['Frequency']['mean'], .75) else 'high' if x>np.quantile(tx_user_viz['Frequency']['mean'], .75) else 'low')
newm= tx_user_viz['Monetary']['mean'].apply(lambda x:'medium' if x>=np.quantile(tx_user_viz['Monetary']['mean'], .25) and
                                            x<=np.quantile(tx_user_viz['Monetary']['mean'], .75) else 'high' if x>np.quantile(tx_user_viz['Monetary']['mean'], .75) else 'low')
new_tx_user=pd.DataFrame([newr,newf,newm],index=['Recency','Frequency','Monetary'])
new_tx_user.T
```

| Segment | Recency | Frequency | Monetary |
|---|---|---|---|
| About to sleep | medium | low | low |
| At risk | high | medium | high |
| Cant loose them | high | high | high |
| Champions | low | high | medium |
| Lost | high | medium | medium |
| Loyal customers | medium | high | high |
| Need attention | medium | medium | medium |
| New customers | low | low | low |
| Potential loyalist | low | medium | medium |
| Promising | medium | low | low |

## Menambahkan Fitur

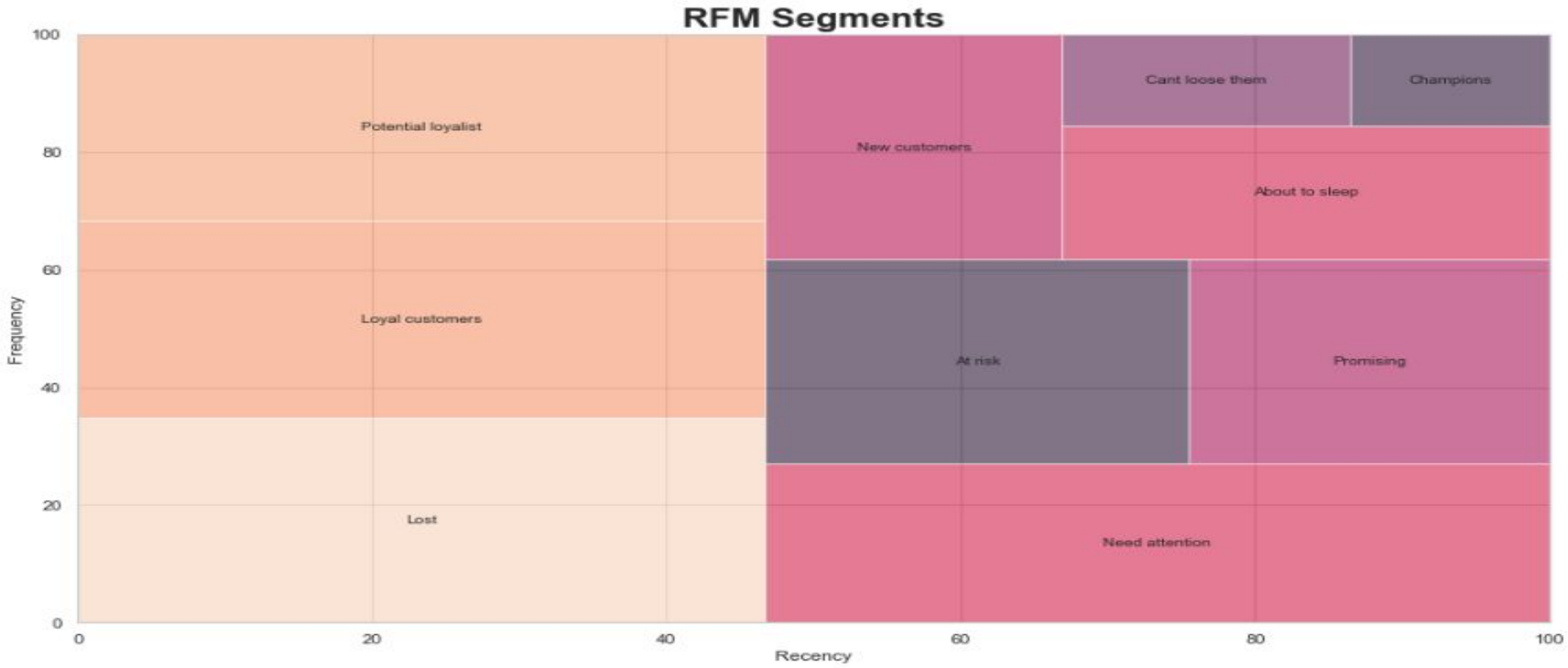
```
#Adding the Clusters feature to the orignal dataframe.
dft  = df.merge(tx_user[['ID', 'Segment']],on='ID')
dft.head()
```

# Clustering With RFM Analysis

Rakamin Academy

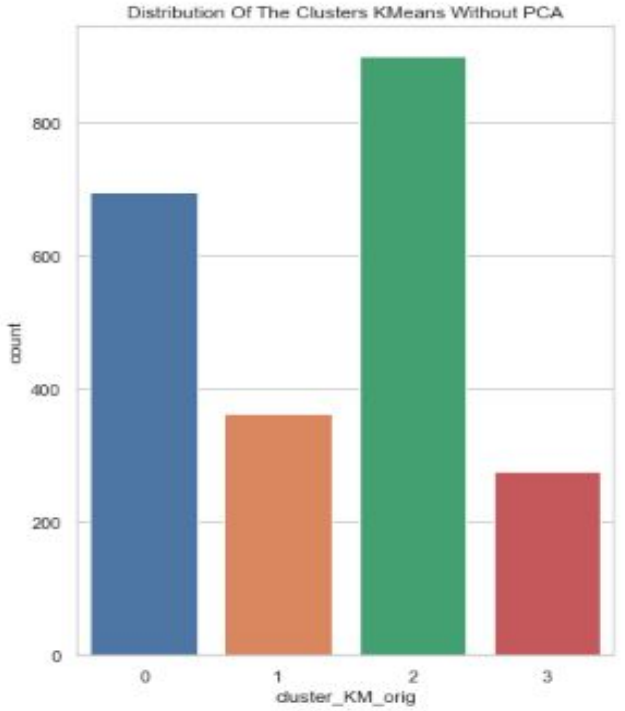
**Segment Visualization**

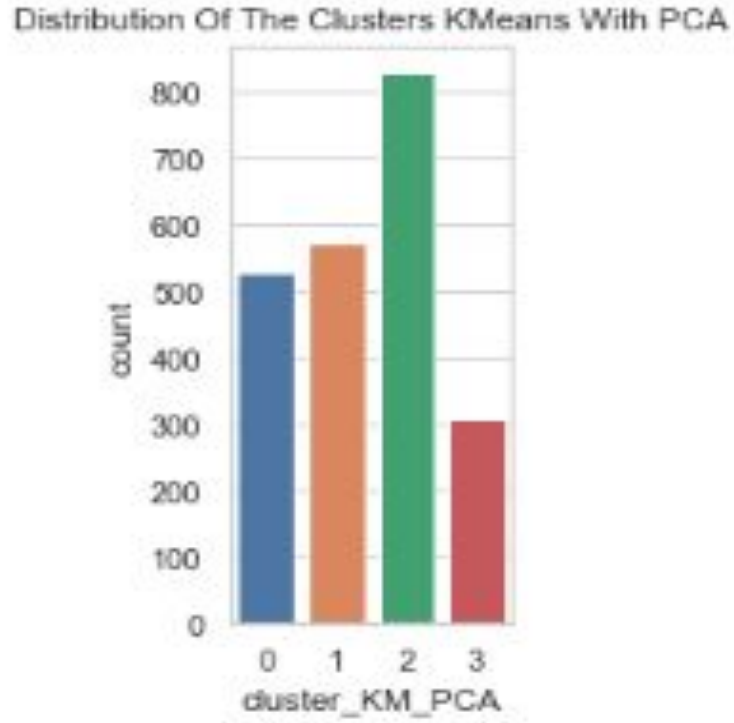
```
import squarify
fig = plt.gcf()
ax = fig.add_subplot()
fig.set_size_inches(16, 9)
squarify.plot(sizes=tx_user['Segment'].value_counts(),
              label=tx_user['Segment'].value_counts().index, alpha=.6 )
plt.title("RFM Segments",fontsize=22,fontweight="bold")
ax.set_xlabel('Recency',fontsize=12)
ax.set_ylabel('Frequency',fontsize=12)
plt.axis('on')
plt.show()
```

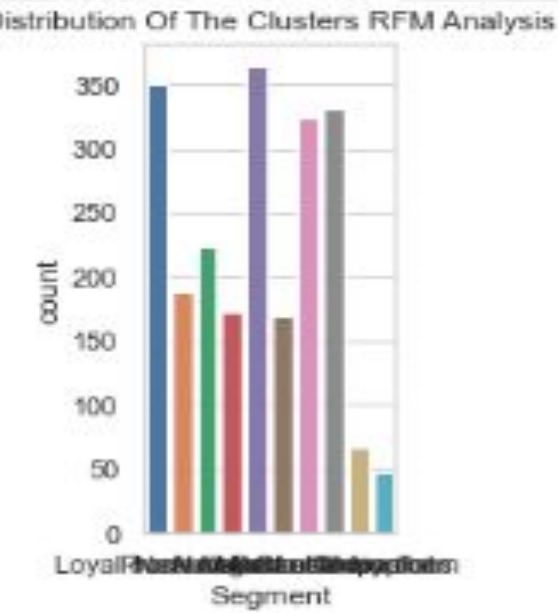


# Evaluation Model

```
#Plotting countplot of clusters
plt.figure(figsize=[20,8])

plt.subplot(1,3,1)
plo = sns.countplot(x=dft["cluster_KM_orig"])
plo.set_title("Distribution Of The Clusters KMeans Without PCA")
plt.show()

plt.subplot(1,3,2)
plp = sns.countplot(x=dft["cluster_KM_PCA"])
plp.set_title("Distribution Of The Clusters KMeans With PCA")
plt.show()

plt.subplot(1,3,3)
plr = sns.countplot(x=dft["Segment"])
plr.set_title("Distribution Of The Clusters RFM Analysis")
plt.show()
```

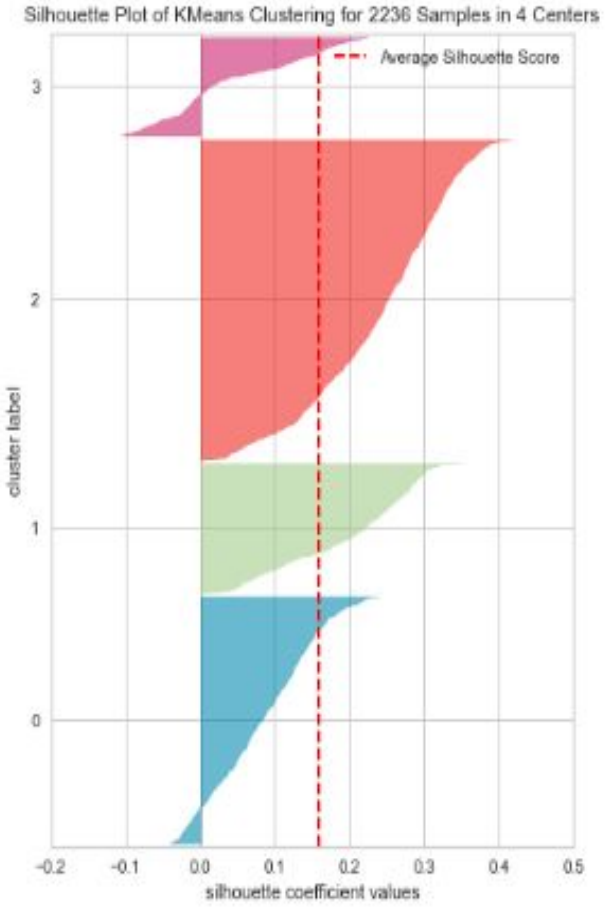


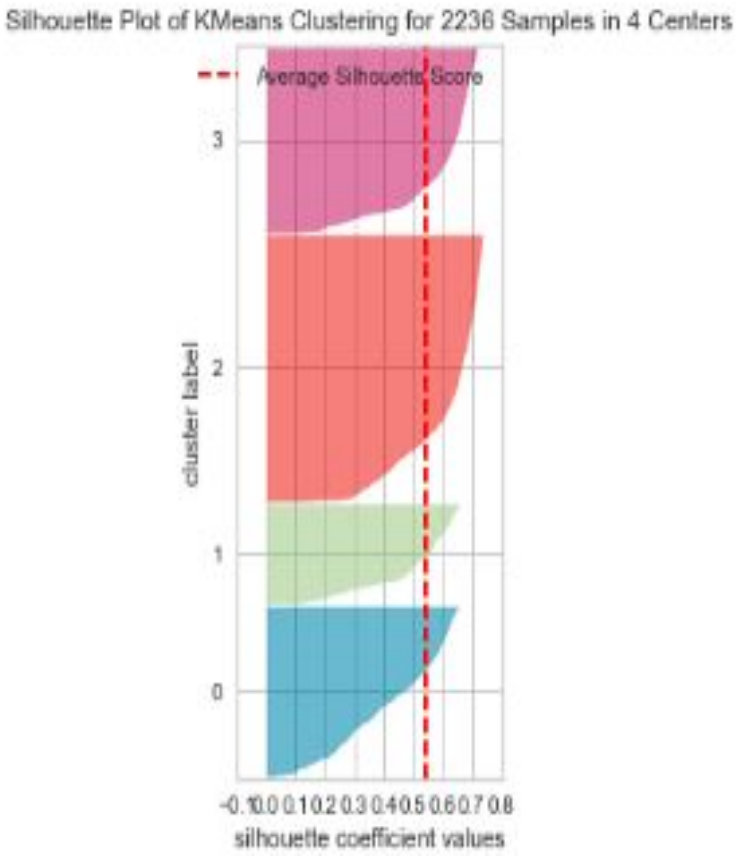



# Silhouette Plot

```
from yellowbrick.cluster import SilhouetteVisualizer
plt.figure(figsize=[20,8])

#silhoutte plot Kmeans without PCA
plt.subplot(1,3,1)
model = KMeans(n_clusters=4, random_state=0)
visualizer = SilhouetteVisualizer(model, colors='yellowbrick')
visualizer.fit(clus_dfsc)
visualizer.show()
plt.show()

#silhoutte plot Kmeans with PCA
plt.subplot(1,3,2)
model = KMeans(n_clusters=4, random_state=0)
visualizer = SilhouetteVisualizer(model, colors='yellowbrick')
visualizer.fit(PCA_ds)
visualizer.show()
plt.show()
```



```
# Visualization KMeans WIthout PCA
newdf = pd.DataFrame(data= df, columns=['Income', 'Has_child', 'Recency', 'MntWines', 'MntFruits',
                 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
                 'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
                 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth', 'Total_Cmp', 'Ever_Accept',
                 'Age'])
hasilcl2 = dft[['ID', 'cluster_KM_orig', 'Income', 'Has_child', 'Recency', 'MntWines', 'MntFruits',
                 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
                 'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
                 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth', 'Total_Cmp', 'Ever_Accept',
                 'Age']].copy()
analysis_res = hasilcl2.groupby('cluster_KM_orig').agg({'mean'})
analysis_res['Banyak_cust'] = hasilcl2.groupby('cluster_KM_orig')['ID'].count()

plt.figure(figsize=(16, 30))
for i, var in enumerate(newdf.columns):
    plt.subplot(8,3,i+1)
    sns.barplot(x = analysis_res.reset_index().cluster_KM_orig, y = analysis_res[var]['mean'])
    plt.ylabel(var, fontsize=15)
    plt.xlabel('Cluster', fontsize=15)
plt.tight_layout()
plt.show()
```

```
# Visualization KMeans With PCA
newdf = pd.DataFrame(data= df, columns=['Income', 'Has_child', 'Recency', 'MntWines', 'MntFruits',
                 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
                 'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
                 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth', 'Total_Cmp', 'Ever_Accept',
                 'Age'])
hasilcl2 = dft[['ID', 'cluster_KM_PCA', 'Income', 'Has_child', 'Recency', 'MntWines', 'MntFruits',
                 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
                 'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
                 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth', 'Total_Cmp', 'Ever_Accept',
                 'Age']].copy()
analysis_res = hasilcl2.groupby('cluster_KM_PCA').agg({'mean'})
analysis_res['Banyak_cust'] = hasilcl2.groupby('cluster_KM_PCA')['ID'].count()

plt.figure(figsize=(16, 30))
for i, var in enumerate(newdf.columns):
    plt.subplot(8,3,i+1)
    sns.barplot(x = analysis_res.reset_index().cluster_KM_PCA, y = analysis_res[var]['mean'])
    plt.ylabel(var, fontsize=15)
    plt.xlabel('Cluster', fontsize=15)
plt.tight_layout()
plt.show()
```

# Business Insight and Recommendation

| Segment | Insight & Recommendation |
|---|---|
| ***Champions*** | • Customer ini memiliki karakteristik: sangat aktif berbelanja, menghabiskan uang cukup banyak, dan intensitas berbelanjanya sangat tinggi.<br>• Karakteristik customer ini sangatlah loyal terhadap perusahaan sehingga perlu diberikan perlakuan istimewa dengan rmemberikan Reward. Contoh: program tebus murah untuk item tertentu menggunakan poin Reward, pengambilan hadiah karena telah membeli dalam jumlah tertentu, dsb. |
| ***Loyal customers*** | • Customer ini memiliki karakteristik: aktif berbelanja, menghabiskan uang sangat banyak, dan intensitas berbelanjanya sangat tinggi.<br>• Meningkatkan keaktifan customer dengan memberikan progam pengumpulan poin yang dapat ditukar dengan item/ promo tertentu jika telah terkumpul poinnya, discount untuk customer yang memiliki member card, dll. |
| ***Potential loyalist*** | • Customer ini memiliki karakteristik: sangat aktif berbelanja, menghabiskan uang cukup banyak, dan intensitas berbelanjanya cukup tinggi.<br>• Meningkatkan frekuensi customer berbelanja seperti sale/discount item tertentu di hari atau tanggal tertentu |
| ***Need attention*** | • Customer ini memiliki karakteristik: cukup aktif berbelanja, menghabiskan uang cukup banyak, dan intensitas berbelanjanya cukup tinggi.<br>• Jenis customer ini butuh perhatian ekstra untuk meningkatkan keaktifannya berbelanja dan intensitas berbelanja seperti membuat promosi bundling item yang paling diminati. Contoh: promo bundling Daging Sapi + Wine |

# Pembagian Tugas

| NAMA | STAGE 0 | STAGE 1 | STAGE 2 | STAGE 3 | STAGE 4 |
|---|---|---|---|---|---|
| **Nur Imam Masri** | Team Leader, Komunikasi Mentor, Git, Problem Statement, Goals, Objectives, Roles, Business Metrics | Multivariate Analysis, Business Insight, Git, Finalisasi Notebook | Feature Selection, Data Split & Imbalance, Additional Method Prep, Git, Finalisasi Notebook | Classification Models, Git, Finalisasi Notebook | PPT : (Modelling Classification), Finalisasi Laporan, Notebook, Notulensi |
| **Astuti Rahmawati** | Problem Statement, Goals, Objectives, Roles, Business Metrics | Univariate Analysis, Business | Feature Selection, Data Split & Imbalance | Clustering Models | PPT : (Business Recommendation & Simulation) |
| **Prasidya Bagaskara** | Problem Statement, Goals, Objectives, Roles, Business Metrics | Multivariate Analysis, Business Insight | Feature Selection, Data Split & Imbalance | Classification Models | PPT : (Data Prep) |
| **Moh. Harwin Prayoga** | Problem Statement, Goals, Objectives, Roles, Business Metrics | Univariate Analysis, Business Insight | Handling Outliers dan Feature Engineering/Extraction | Classification Models | PPT : (EDA) |
| **Riskiyatul Hasanah** | Problem Statement, Goals, Objectives, Roles, Business Metrics | Descriptive Statistics, Business Insight | Handling Missing Value, Duplicated Values and Invalid Values | Classification Models | PPT : (Problem Statement) |

| NAMA | STAGE 0 | STAGE 1 | STAGE 2 | STAGE 3 | STAGE 4 |
|---|---|---|---|---|---|
| **M Rayhan Azzindani** | Problem Statement, Goals, Objectives, Roles, Business Metrics | Descriptive Statistics, Business Insight | Handling Outliers dan Feature Engineering/Extraction | Clustering Models, Finalisasi Notebook | PPT : (Modelling Clustering) |
| **Siti Hajjah Mardiah** | Problem Statement, Goals, Objectives, Roles, Business Metrics | Univariate Analysis, Business Insight | Feature Transformation dan Feature Encoding | Classification Models | PPT : (Business Recommendation & Simulation) |
| **Christine** | Problem Statement, Goals, Objectives, Roles, Business Metrics | Multivariate Analysis, Business Insight | Feature Transformation dan Feature Encoding | Clustering Models | PPT : (EDA) |
| **M. Ifzal Asril** | Problem Statement, Goals, Objectives, Roles, Business Metrics | Descriptive Statistics, Business Insight | Handling Missing Value, Duplicated Values and Invalid Values | Clustering Models | PPT : (Data Understanding) |

# Pembagian Tugas

**Note :**

- Semua akan mengerjakan **Laporan** berdasarkan pembagian tugas di table sebelumnya
- Hasil pengerjaan akan **digabungkan dan di finalisasi**

**Pembagian Notulensi Mentoring :**

**Stage 0**

M Rayhan Azzindani

**Stage 1**

Astuti Rahmawati

M. Ifzal Asril

**Stage 2**

Siti Hajjah Mardiah

Christine

**Stage 3**

Riskiyatul Hasanah

Prasidya Bagaskara

**Stage 4**

Nur Imam Masri

Moh. Harwin Prayoga

Rakamin Academy

# END