BA715\_ Capstone Project| Exploration & Modeling  
Centennial College  
From: Nguyen Vuong Hoai Vu (301131497)  
For: Prof. David Parent  
Prof. Mustafa Ahmed  
Prof: Resmi  
  
Due date: Dec. 16th, 2021



Contents

[INTRODUCTION 3](#_Toc90628236)

[1. Background 3](#_Toc90628237)

[2. Data set introduction 4](#_Toc90628238)

[DATA EXPLORATION 6](#_Toc90628239)

[3. Data importing and joining 6](#_Toc90628240)

[4. Data Exploration and Engineering 10](#_Toc90628241)

[MODELLING 24](#_Toc90628242)

[5. Cluster Analysis 24](#_Toc90628243)

[6. Logistic Regression 27](#_Toc90628244)

[7. Decision Tree 29](#_Toc90628245)

[8. Gradient Boosting Classifier 35](#_Toc90628246)

[9. Neural Network 38](#_Toc90628247)

[10. Model Comparison 39](#_Toc90628248)

[11. Find the Significant Features when applying Logistic Regression 40](#_Toc90628249)

[CONCLUSION 44](#_Toc90628250)

[GOVERNANCE PLAN 51](#_Toc90628251)

[12. Risks Management 51](#_Toc90628252)

[a. Input 51](#_Toc90628253)

[b. . Algorithms 52](#_Toc90628254)

[c. . Output 54](#_Toc90628255)

[13. Variables level monitoring 55](#_Toc90628256)

# INTRODUCTION

## Background

I create this report as the Capstone Project for the course **BA715 in the program Business Analytics and Insights on December 04th, 2021**. This report aims to analyze the data set called “Marketing Data” (Daoud, 2020) using different types of analysis techniques and models that have been acquired during the program. As a result, I try to find the best fit model to predict the success level of the next marketing campaign based on the historical data and give insightful recommendations from the findings. All the analyses are completed using MySQL and Python. This report is for academic purposes only.

Business background: the company ABC is trying to launch a new marketing campaign that supposes to happen in the next coming holiday season. The marketing director is asking me as a market insights analyst to predict the success rate of the next campaign and recommend for improvements in the acceptance rate of the next campaign based on the 2-year historical data

The Business Objective is to increase sales by direct improvement of the marketing campaign’s acceptance. Besides prediction, this report tries to address what factors are currently impacting the success rate of a marketing campaign, who are likely to accept the new campaign and which products they would tend to buy… and tries to recommend what needs focusing on in the next campaign.

## Data set introduction

This data set has been completed in 2020 to measure the profiles of customers, amount of their purchased products, channels of purchasing and the acceptances of 6 different recently marketing campaigns during the last 2 years, including one last campaign which is used as the target variable in this analysis report. This data set includes 3 tables with 2,240 rows for each table that represent 2,240 unique customer data surveys:

* marketing\_data\_profile: it contains all profiles of 2240 unique customers who have been purchasing our products in the last 2 years.
* marketing\_data\_purchases: it contains the amounts of 6 products that customers have purchased and the 5 channels that they have purchased from, during the last 2 years.
* marketing\_data\_responses: it contains the acceptance from each customer toward our 5 marketing campaigns and whether they have complained about our product or not during the last 2 years.

The detailed information is shown in Table 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Label** | **Role** | **Description** | **Code** | **Type** |
| **marketing\_data\_profiles** | | | | | |
| 1 | ID | ID | Customer's unique identifier |  | Int |
| 2 | Year\_Birth | N/A | Customer's birth year |  | Int |
| 3 | Education | Input | Customer's education level | Graduation, PhD, Master, | Cat |
| 4 | Marital\_Status | Input | Customer's marital status | Married, Together, Single, … | Cat |
| 5 | Income | Input | Customer's yearly household income |  | $ (catl) |
| 6 | Kidhome | Input | Number of children in customer's household |  | Int |
| 7 | Teenhome | Input | Number of teenagers in customer's household |  | Int |
| 8 | Dt\_Customer | N/A | Date of customer's enrollment with the company |  | Date (catl) |
| 9 | Recency | Input | Number of days since customer's last purchase |  | Int |
| 10 | Country | Input | Customer's location | SA, CA, AUS,… | Cat |
| **marketing\_data\_purchases** | | | | | |
| 11 | ID\_Cust | ID | Customer's unique identifier |  | Int |
| 12 | MntWines | Input | Amount spent on wine in the last 2 years |  | Int |
| 13 | MntFruits | Input | Amount spent on fruits in the last 2 years |  | Int |
| 14 | MntMeatProducts | Input | Amount spent on meat in the last 2 years |  | Int |
| 15 | MntFishProducts | Input | Amount spent on fish in the last 2 years |  | Int |
| 16 | MntSweetProducts | Input | Amount spent on sweets in the last 2 years |  | Int |
| 17 | MntGoldProds | Input | Amount spent on gold in the last 2 years |  | Int |
| 18 | NumDealsPurchases | Input | Number of purchases made with a discount |  | Int |
| 19 | NumWebPurchases | Input | Number of purchases made through the company's website |  | Int |
| 20 | NumCatalogPurchases | Input | Number of purchases made using a catalogue |  | Int |
| 21 | NumStorePurchases | Input | Number of purchases made directly in stores |  | Int |
| 22 | NumWebVisitsMonth | Input | Number of visits to company's web site in the last month |  | Int |
| **marketing\_data\_responses** | | | | | |
| 23 | ID\_Cust | ID | Customer's unique identifier |  | Int |
| 24 | AcceptedCmp3 | Input | 1 if customer accepted the offer in the 3rd campaign, 0 otherwise |  | Int |
| 25 | AcceptedCmp4 | Input | 1 if customer accepted the offer in the 4th campaign, 0 otherwise |  | Int |
| 26 | AcceptedCmp5 | Input | 1 if customer accepted the offer in the 5th campaign, 0 otherwise |  | Int |
| 27 | AcceptedCmp1 | Input | 1 if customer accepted the offer in the 1st campaign, 0 otherwise |  | Int |
| 28 | AcceptedCmp2 | Input | 1 if customer accepted the offer in the 2nd campaign, 0 otherwise |  | Int |
| 29 | Response | Target | 1 if customer accepted the offer in the last campaign, 0 otherwise |  | Int |
| 30 | Complain | Input | 1 if customer complained in the last 2 years, 0 otherwise |  | Int |
| 31 | Country | Input | Customer's location | SA, CA, AUS,… | Cat |

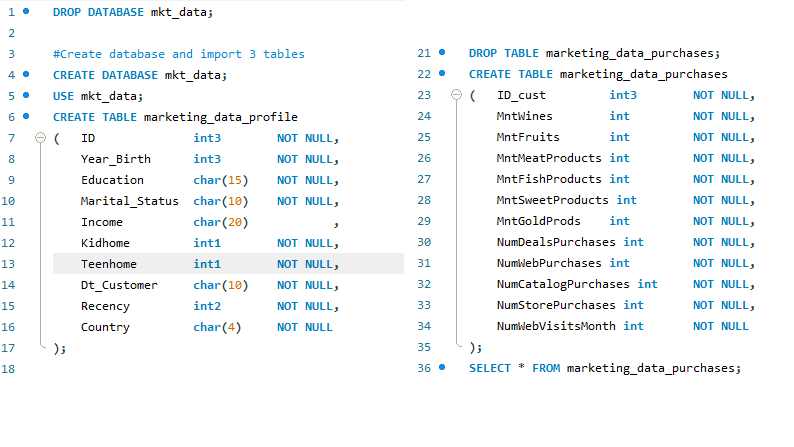
Table 1: List of data variables and details.

# DATA EXPLORATION

## Data importing and joining

The first thing is to import 3 tables marketing\_data\_profiles, marketing\_data\_purchases, and marketing\_data\_responses into MySQL then try to join the 3 tables together with the key is the ID of each unique customer.

Firstly, I created the database named mkt\_data and created 3 empty tables with the corresponding columns in MySQL.



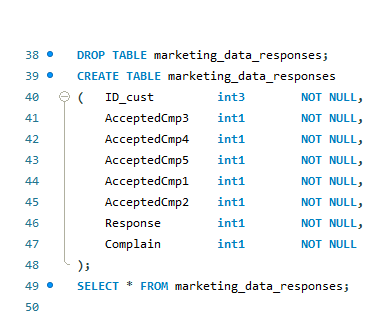
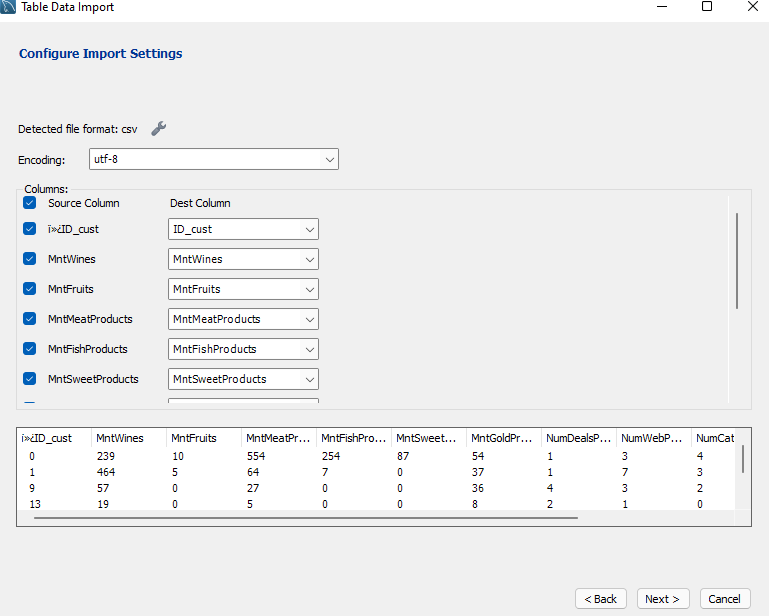
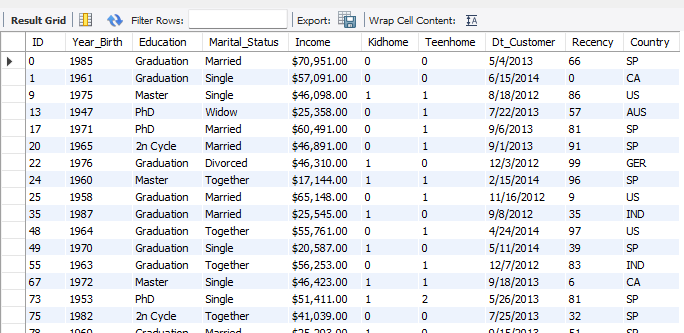
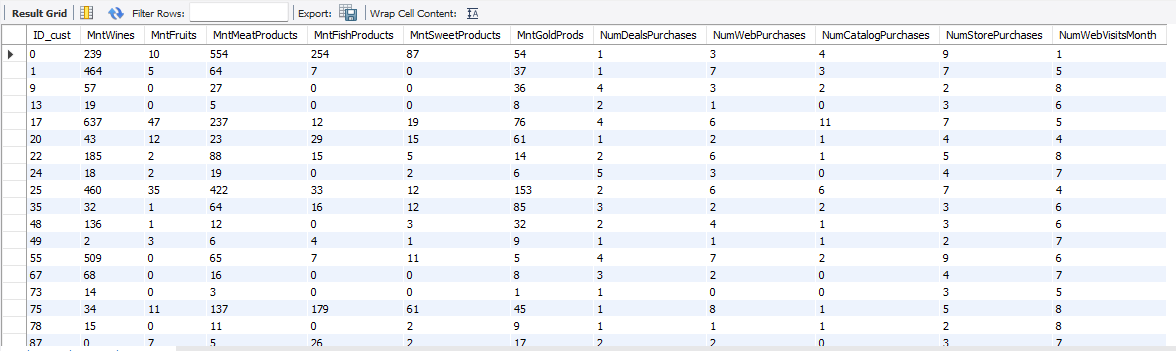


Figure 1: Creating 3 empty tables to import data to MySQL

Then I imported 3 tables from \*.csv files into 3 empty tables:







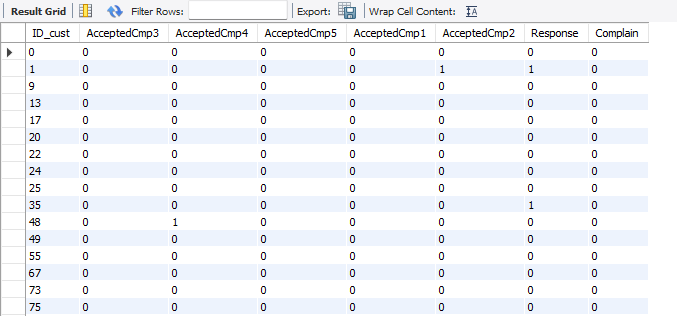
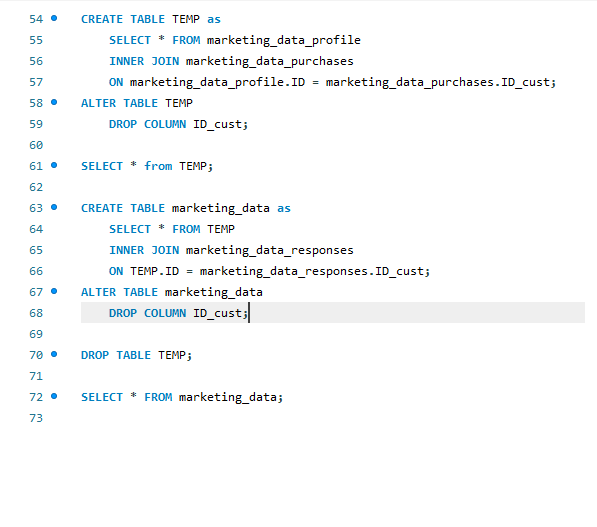


Figure 2: Importing data into 3 empty tables

I tried to merge the 3 imported tables into the marketing\_data table:



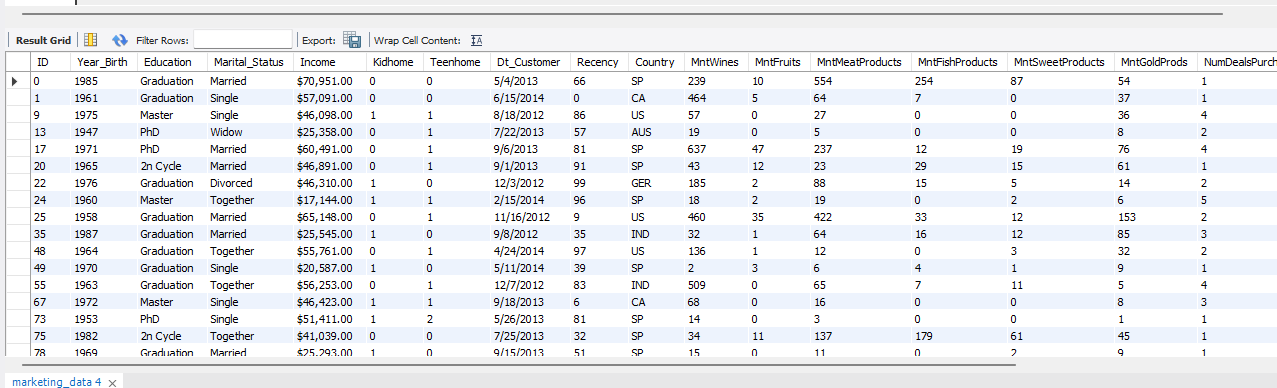


Figure 3: Merging 3 tables into 1 table

Finally, I exported the merged table with 28 columns and 2,240 observations as the marketing\_data.csv file as the output.

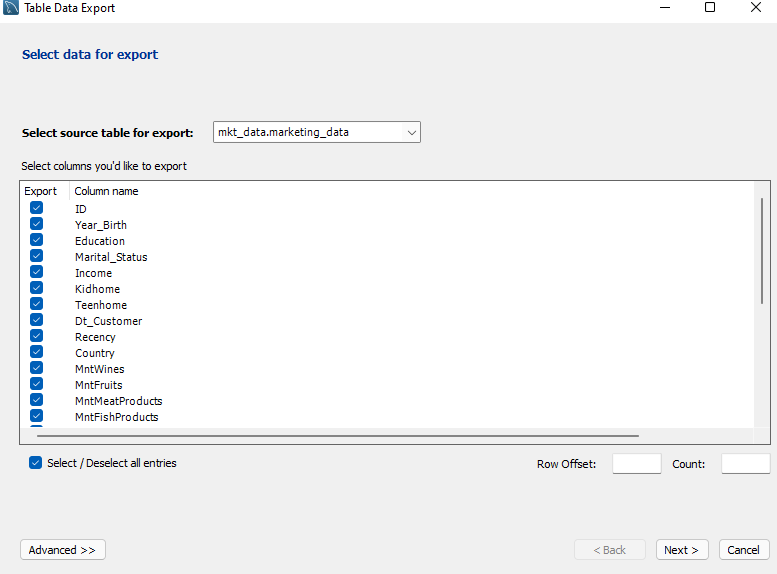


Figure 4: Exporting final data table for further analysis

## Data Exploration and Engineering

For data exploration, I imported the full table into Google Colab to implement further analysis using Python

#Import libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.cluster import KMeans

from sklearn.preprocessing import LabelEncoder

%matplotlib inline

#Read and explore the dataset

url = '…/1ZxD4eAR1DK\_2ts0u4uxZ1bOtIMnrJ3JJ/view?usp=sharing'

file\_id = url.split('/')[-2]

dwn\_url='https://drive.google.com/uc?id=' + file\_id

mktdata = pd.read\_csv(dwn\_url)

print(mktdata.info())

mktdata.head()

Output:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2240 entries, 0 to 2239

Data columns (total 28 columns):

# 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 object

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 Response 2240 non-null int64

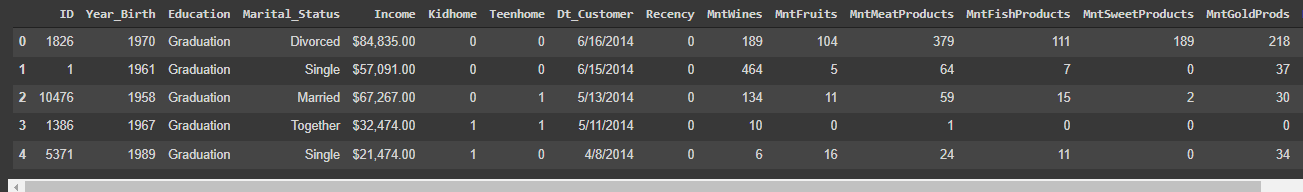
26 Complain 2240 non-null int64

27 Country 2240 non-null object

dtypes: int64(23), object(5)

memory usage: 490.1+ KB

None



Code 1: Data importing and overview

I noticed some points that need to be touched up and cleaned before doing any analysis:

1. The “ Income “ variable has some white space, it is stored as a categorical variable that is not suitable for a currency variable and it has some missing values:

#Noticed Income variable has 24 missing values, I will try to explore the variable to see which method will be used to impuse

features = ['Income']

sns.set\_style('dark')

for col in features:

    plt.figure(figsize=(25,4))

    plt.subplot(131)

    sns.distplot(mktdata\_E[col], label="skew: " + str(np.round(mktdata\_E[col].skew(),2)))

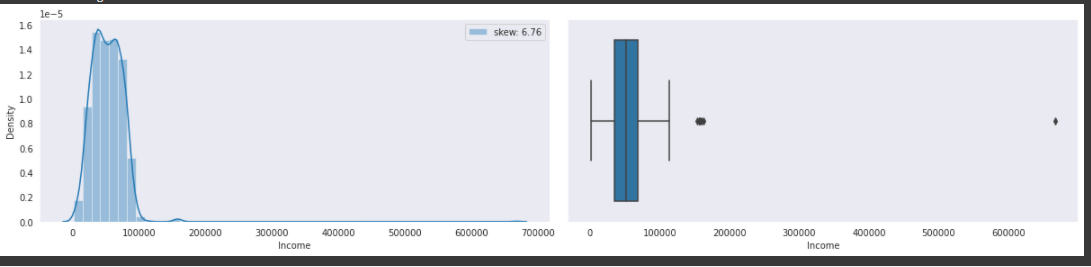
    plt.legend()

    plt.subplot(132)

    sns.boxplot(mktdata\_E[col])

    plt.tight\_layout()

    plt.show()



Code 2: Exploring “Income” variable

I saw some outliners appear so it is better to apply the median imputation instead of the mean imputation.

#impute missing values in Income column with the median value

Mktdata\_E['Income'] = mktdata\_E['Income'].fillna(mktdata\_E['Income'].median())

Mktdata\_E.info()

Output:

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2240 entries, 0 to 2239

Data columns (total 28 columns):

# 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 2240 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 Response 2240 non-null int64

26 Complain 2240 non-null int64

27 Country 2240 non-null object

dtypes: float64(1), int64(23), object(4)

memory usage: 490.1+ KB

Code 3: Applying median imputation to missing values in the “Income” column

1. “Dt\_Customer” is stored as a categorical type which is supposed to be a DateTime type. I also want to transform this variable into “Year\_as\_Cust” which contains the number of years as a customer. It would make more sense when applying prediction modelling.

# The column Dt\_Customer should be stored as date and time type

mktdata\_E['Dt\_Customer'] = mktdata\_E['Dt\_Customer'].astype('datetime64[ns]')

# Calculate the number of year as a customer for each row and store in a new column

from datetime import datetime

mktdata\_E['Years\_as\_Cust'] = pd.DatetimeIndex(mktdata\_E['Dt\_Customer']).year

mktdata\_E['Years\_as\_Cust'] = 2020 - mktdata\_E['Years\_as\_Cust']

print(mktdata['Years\_as\_Cust'].describe())

Code 4: Engineering “Years\_as\_Cust” column

1. I want to transform “Year\_Birth” into “Ages” which makes more sense when applying prediction modelling.

# Calculate the ages of customers

mktdata\_E['Ages'] = 2020 - mktdata\_E['Year\_Birth']

print(mktdata\_E['Ages'].describe())

Code 5: Engineering “Age” column

1. I want to engineer the total amount of purchased products and the total number of purchases to see if they are significantly contributing to the target outcomes.

# Calculate total amount of product purchased and save them into 'MntTotal'

columns\_Mnt = ['MntFruits', 'MntMeatProducts', 'MntFishProducts',

       'MntSweetProducts', 'MntGoldProds']

mktdata\_E['MntTotal'] = mktdata\_E[columns\_Mnt].sum(axis=1)

# Calculate total purchases and save them into 'TotalPurchases'

columns\_pur = ['NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases']

mktdata\_E['TotalPurchases'] = mktdata\_E[columns\_pur].sum(axis=1)

Code 6: Engineering “MntTotal” and “TotalPurchases” column

1. I also want to create indicators for the customers that have children and the customers that have purchased a deal.

# Assign new variable "ChildHome" with '0' means no children and '1' means having children

def f(row):

    if row['Kidhome'] + row['Teenhome'] > 0:

        val = 1

    else:

        val = 0

    return val

mktdata\_E['ChildHome'] = mktdata\_E.apply(f, axis=1)

# Assign a new variable "UsedDeal": '0' means Haven't purchased any deal, '1' means have purchased at least 1 deal

def d(row):

    if row['NumDealsPurchases'] > 0:

        val = 1

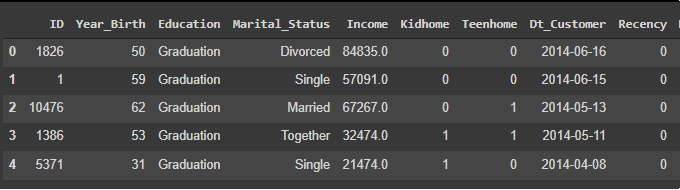
    else:

        val = 0

    return val

mktdata\_E['UsedDeal'] = mktdata\_E.apply(d, axis=1)

mktdata.head()

Output:



Code 7: Engineering indicator for the household with child and customer with deal purchase column

1. I tried to identify outliers and skewness by using distribution histograms and boxplots.

# Select which columns to plot, I eliminate all categorical variables and binary variables

col\_to\_plot = ['Year\_Birth', 'Income',

       'Kidhome', 'Teenhome', 'Recency', 'MntWines',

       'MntFruits', 'MntMeatProducts', 'MntFishProducts',

       'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',

       'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',

       'NumWebVisitsMonth',  'Ages', 'MntTotal',

       'TotalPurchases']

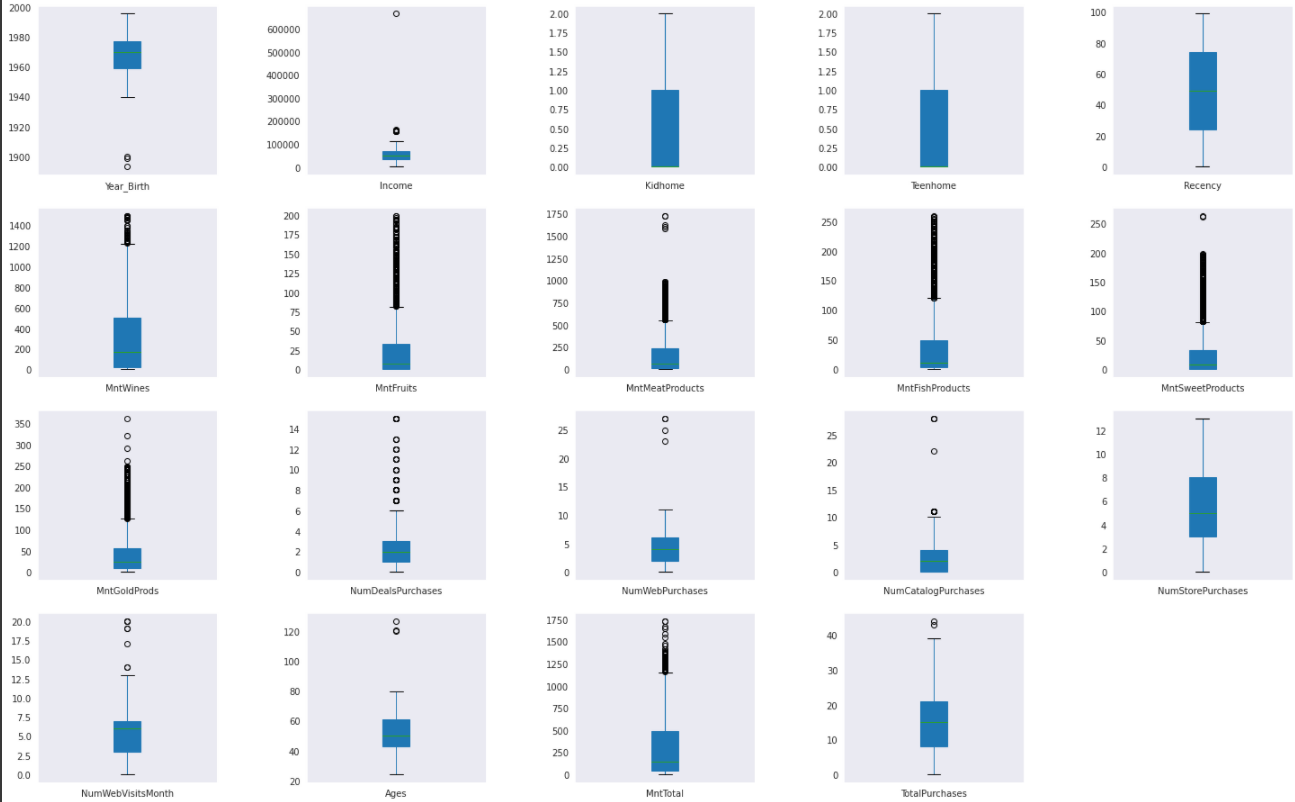
# subplots

sns.set\_style('dark')

mktdata\_E[col\_to\_plot].plot(subplots=True, layout=(5,5), kind='box', figsize=(25,20), patch\_artist=True)

plt.subplots\_adjust(wspace=0.5);

Output:



from scipy.stats import skew

def dist\_plot(df, cols):

  sns.set\_style('dark')

  plt.figure(figsize=(25,20))

  for i, col in enumerate(cols):

    ax = plt.subplot(5,6, i+1)

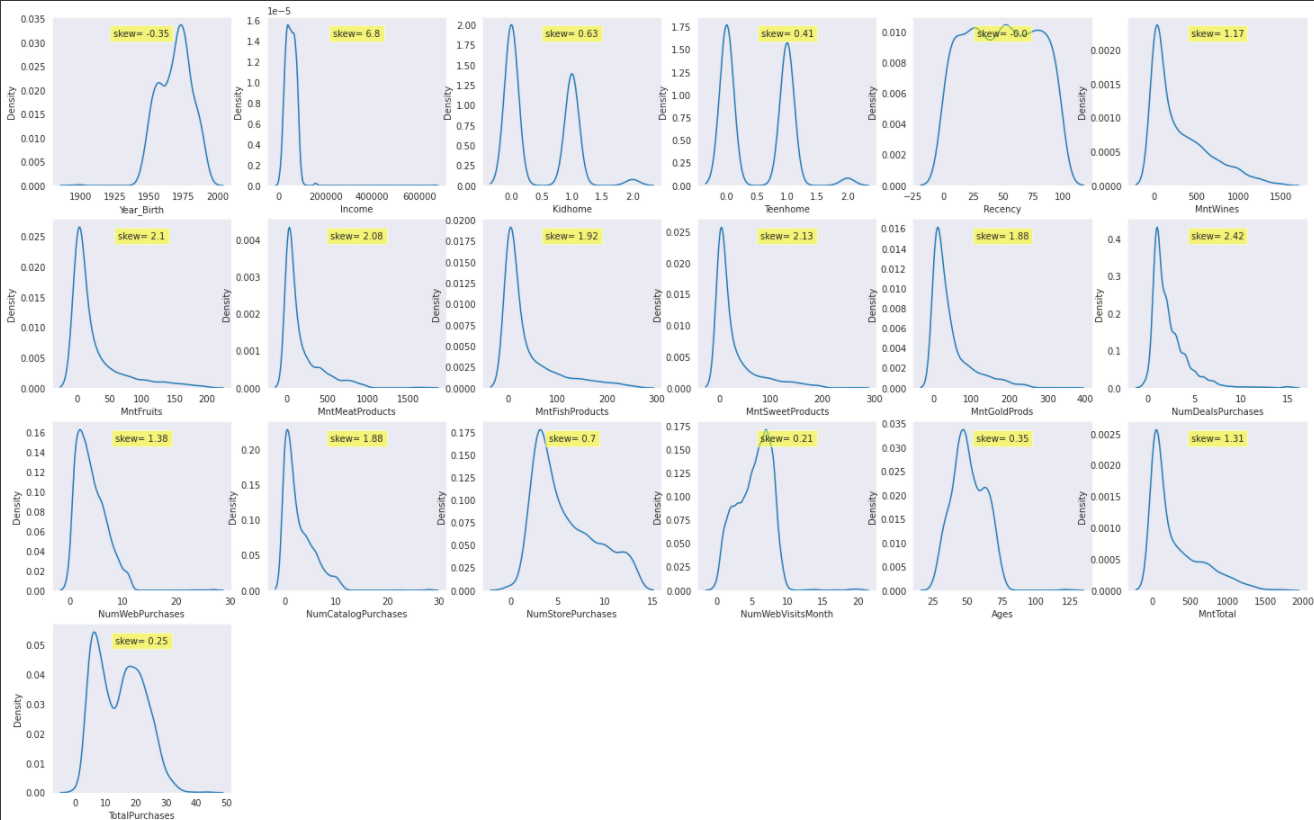
    sns.kdeplot(df[col], ax=ax)

    plt.text(0.5,0.9,"skew= " + str(round(skew(df[col]),2)), bbox=dict(facecolor='yellow', alpha=0.5), horizontalalignment='center', verticalalignment='center', transform=ax.transAxes)

    plt.xlabel(col)

 plt.show()

dist\_plot(mktdata\_E,col\_to\_plot)



As the charts show, the dataset has many outliers and skewness so I applied cap & floor to remove outliers and standard scaler to reduce skewness and bring variables close to normal distribution.

# Save the dataset into new DataFrame called mktdata\_cap and continue with cap & floor the outliers

mktdata\_cap = mktdata\_E

features = ['Year\_Birth', 'Income', 'Recency', 'MntWines',

       'MntFruits', 'MntMeatProducts', 'MntFishProducts',

       'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',

       'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',

       'NumWebVisitsMonth', 'Ages', 'MntTotal',

       'TotalPurchases']

def iqr\_capping(df, cols, factor):

    for col in cols:

        q1 = df[col].quantile(0.25)

        q3 = df[col].quantile(0.75)

        iqr = q3 - q1

        upper\_whisker = q3 + (factor\*iqr)

        lower\_whisker = q1 - (factor\*iqr)

        df[col] = np.where(df[col]>upper\_whisker, upper\_whisker,

                 np.where(df[col]<lower\_whisker, lower\_whisker, df[col]))

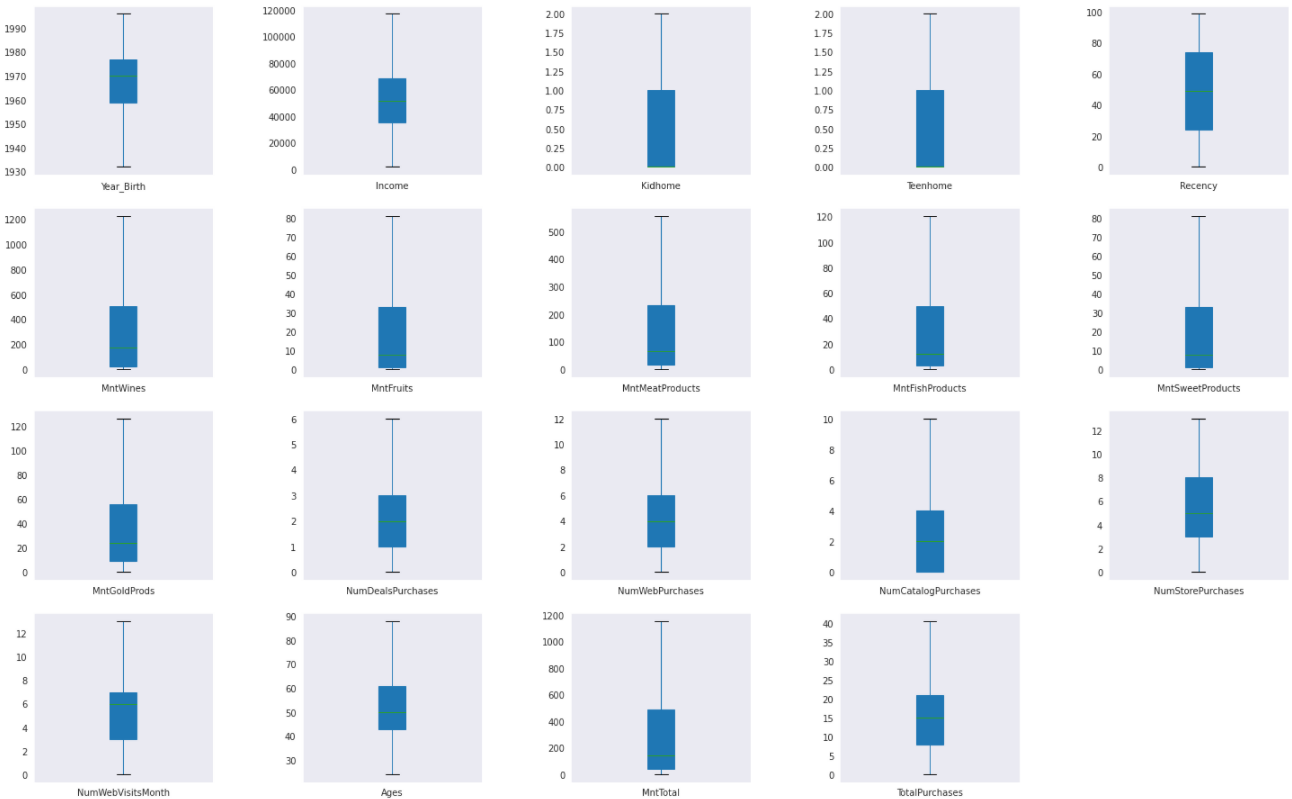
iqr\_capping(mktdata\_cap, features, 1.5)

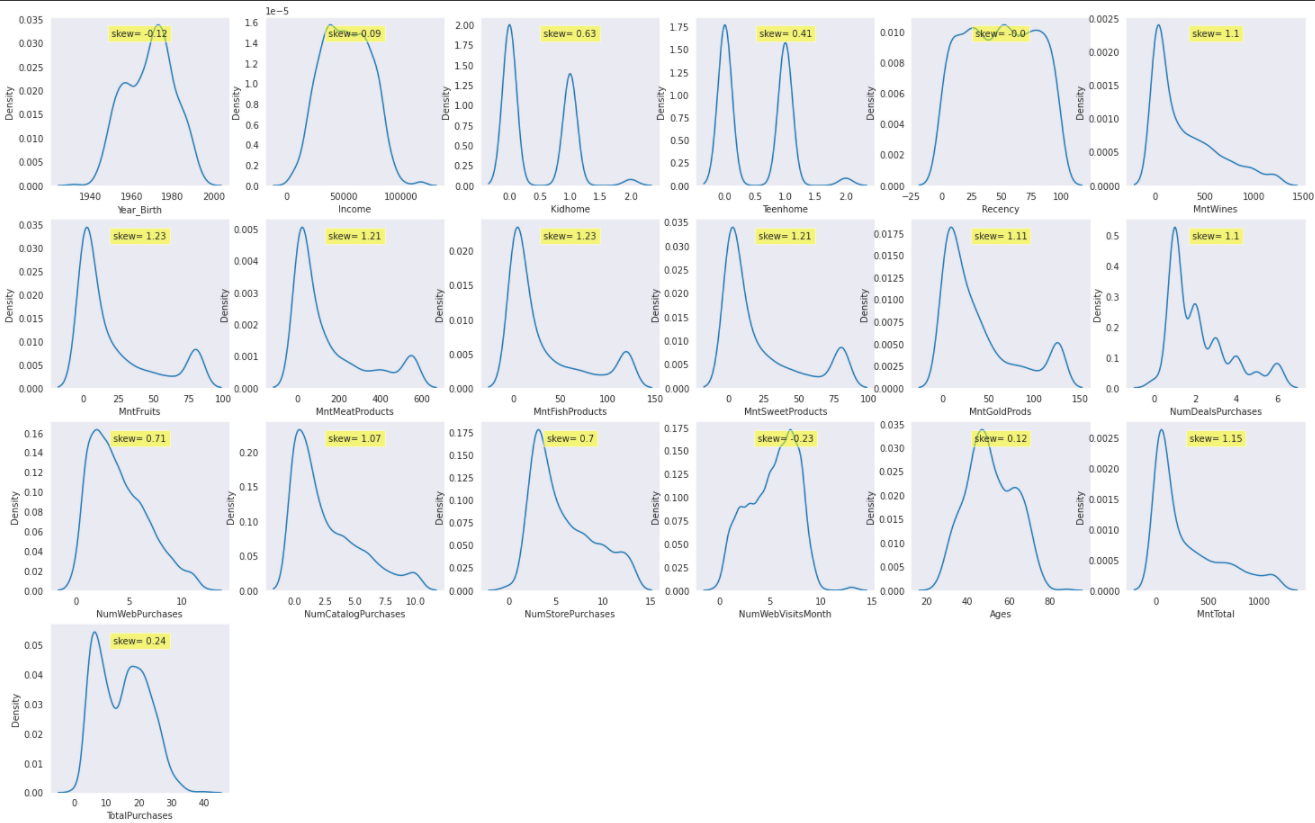
sns.set\_style('dark')

mktdata\_cap[col\_to\_plot].plot(subplots=True, layout=(5,5), kind='box', figsize=(25,20), patch\_artist=True)

plt.subplots\_adjust(wspace=0.5);

dist\_plot(mktdata\_cap, col\_to\_plot)





Code 8: Apply cap & floor transformation for all outliers

All the variables are looking cleaner and less biased but I planned to apply Decision Tree, Logistic Regression, Gradient Boosting, Neural Network… models to predict the target of acceptance toward new marketing campaign, these models use gradient descent as an optimization technique so I need to bring all variables to the same scale range for these models to be easier optimized. I applied the Standard Scaler technique for the list of below variables:

'Year\_Birth', 'Income', 'Recency', 'MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth', 'Ages', 'MntTotal', 'TotalPurchases', 'Years\_as\_Cust'.

Other variables are either categorical variables or in the needed scale (0,1,2)

#Standardize skewed variables

from sklearn.preprocessing import StandardScaler

columns\_scaled = ['Year\_Birth', 'Income', 'Recency', 'MntWines',

       'MntFruits', 'MntMeatProducts', 'MntFishProducts',

       'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',

       'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases',

       'NumWebVisitsMonth',  'Ages', 'MntTotal',

       'TotalPurchases', 'Years\_as\_Cust']

scalar = StandardScaler()

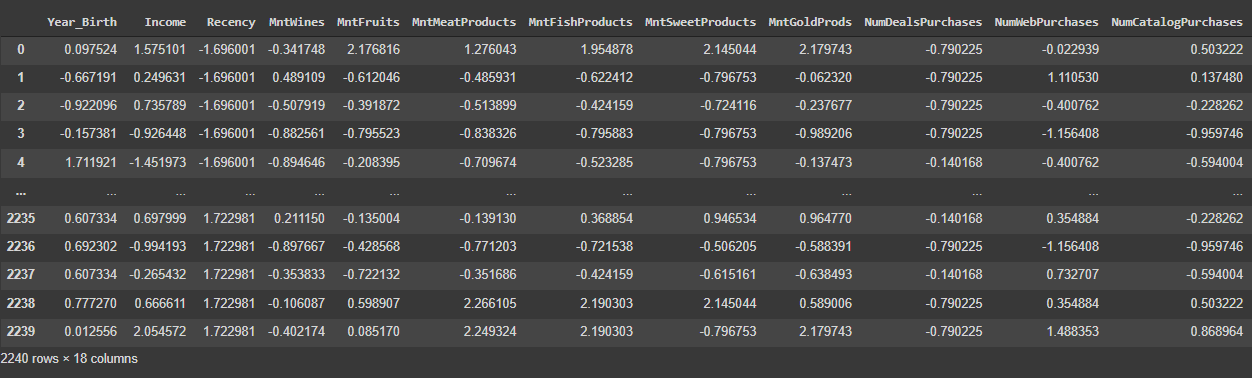
mktdata\_scaled = pd.DataFrame(scalar.fit\_transform(mktdata\_cap[columns\_scaled]), columns=columns\_scaled, index=mktdata\_cap.index)

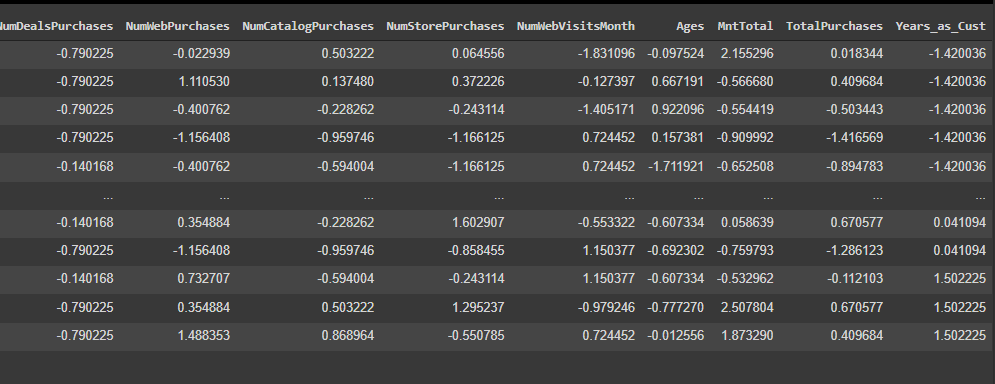
#Adding scaled columns back to dataset

mktdata\_scaled = pd.concat([mktdata\_scaled, mktdata\_cap.drop(axis=1, columns=columns\_scaled)],axis=1)

mktdata\_scaled

Output:





Code 9: Data scaling using Standard Scaler

1. Next, I transformed the categorical variables into dummies variables as below:

# Encoding categorical variables using pandas

cat\_columns = mktdata\_scaled.select\_dtypes(exclude = np.number)

print("Number of unique values per categorical feature:\n", cat\_columns.nunique())

dummies\_cat = pd.get\_dummies(cat\_columns)

# Add to mktdat\_scaled Dataframe and changed the name as mktdata\_trf

mktdata\_trf = pd.concat([mktdata\_scaled,dummies\_cat], axis=1)

mktdata\_trf = mktdata\_trf.drop(columns=cat\_columns.columns)

mktdata\_trf = mktdata\_trf.drop(['ID'], axis=1)

mktdata\_trf.info()

dist\_plot(mktdata\_trf,col\_to\_plot)

Output:

Data columns (total 50 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Year\_Birth 2240 non-null float64

1 Income 2240 non-null float64

2 Recency 2240 non-null float64

3 MntWines 2240 non-null float64

4 MntFruits 2240 non-null float64

5 MntMeatProducts 2240 non-null float64

6 MntFishProducts 2240 non-null float64

7 MntSweetProducts 2240 non-null float64

8 MntGoldProds 2240 non-null float64

9 NumDealsPurchases 2240 non-null float64

10 NumWebPurchases 2240 non-null float64

11 NumCatalogPurchases 2240 non-null float64

12 NumStorePurchases 2240 non-null float64

13 NumWebVisitsMonth 2240 non-null float64

14 Ages 2240 non-null float64

15 MntTotal 2240 non-null float64

16 TotalPurchases 2240 non-null float64

17 Years\_as\_Cust 2240 non-null float64

18 Kidhome 2240 non-null int64

19 Teenhome 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 Response 2240 non-null int64

26 Complain 2240 non-null int64

27 ChildHome 2240 non-null int64

28 UsedDeal 2240 non-null int64

29 Education\_2n Cycle 2240 non-null uint8

30 Education\_Basic 2240 non-null uint8

31 Education\_Graduation 2240 non-null uint8

32 Education\_Master 2240 non-null uint8

33 Education\_PhD 2240 non-null uint8

34 Marital\_Status\_Absurd 2240 non-null uint8

35 Marital\_Status\_Alone 2240 non-null uint8

36 Marital\_Status\_Divorced 2240 non-null uint8

37 Marital\_Status\_Married 2240 non-null uint8

38 Marital\_Status\_Single 2240 non-null uint8

39 Marital\_Status\_Together 2240 non-null uint8

40 Marital\_Status\_Widow 2240 non-null uint8

41 Marital\_Status\_YOLO 2240 non-null uint8

42 Country\_AUS 2240 non-null uint8

43 Country\_CA 2240 non-null uint8

44 Country\_GER 2240 non-null uint8

45 Country\_IND 2240 non-null uint8

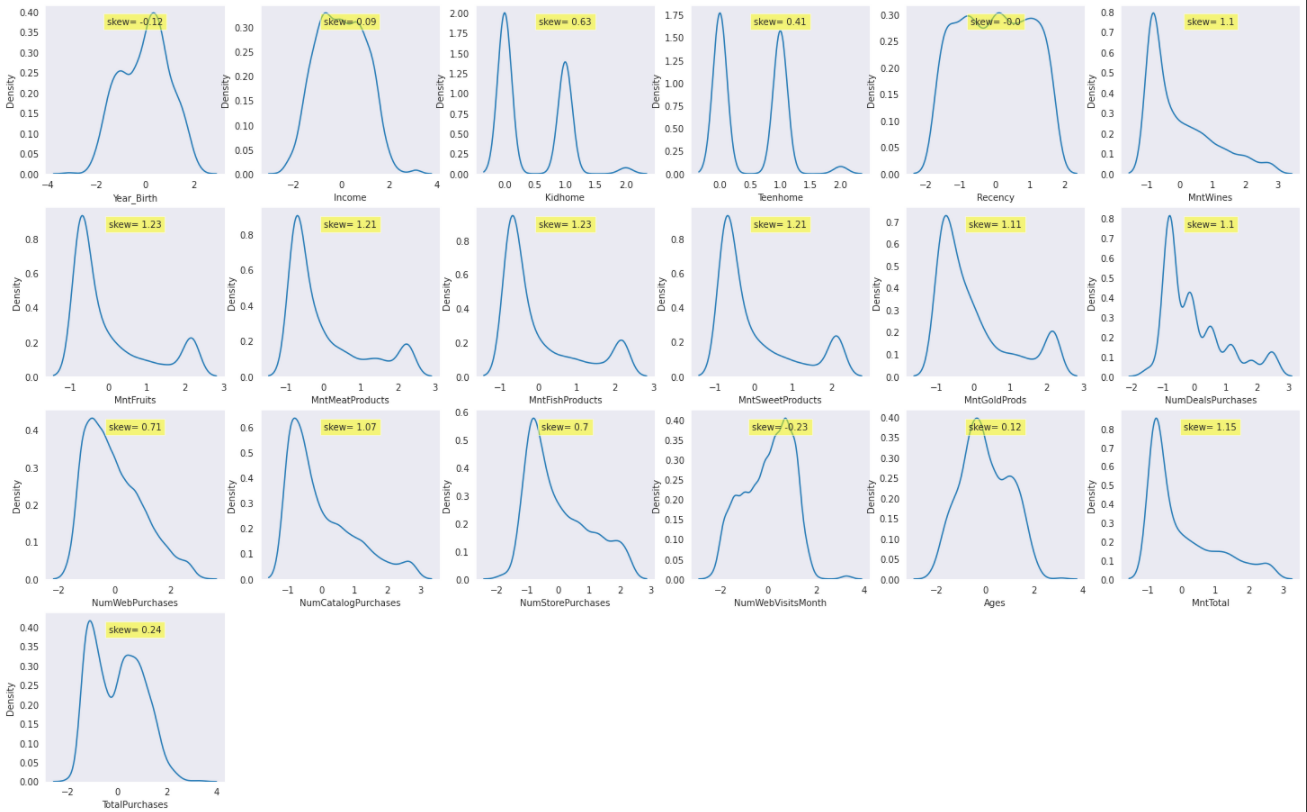
46 Country\_ME 2240 non-null uint8

47 Country\_SA 2240 non-null uint8

48 Country\_SP 2240 non-null uint8

49 Country\_US 2240 non-null uint8

dtypes: float64(18), int64(11), uint8(21)



Code 10: Transforming categorical variables into dummies variables

Look at the final dataset, it was cleaned and scaled to standard distribution and ready to apply in predictive models.

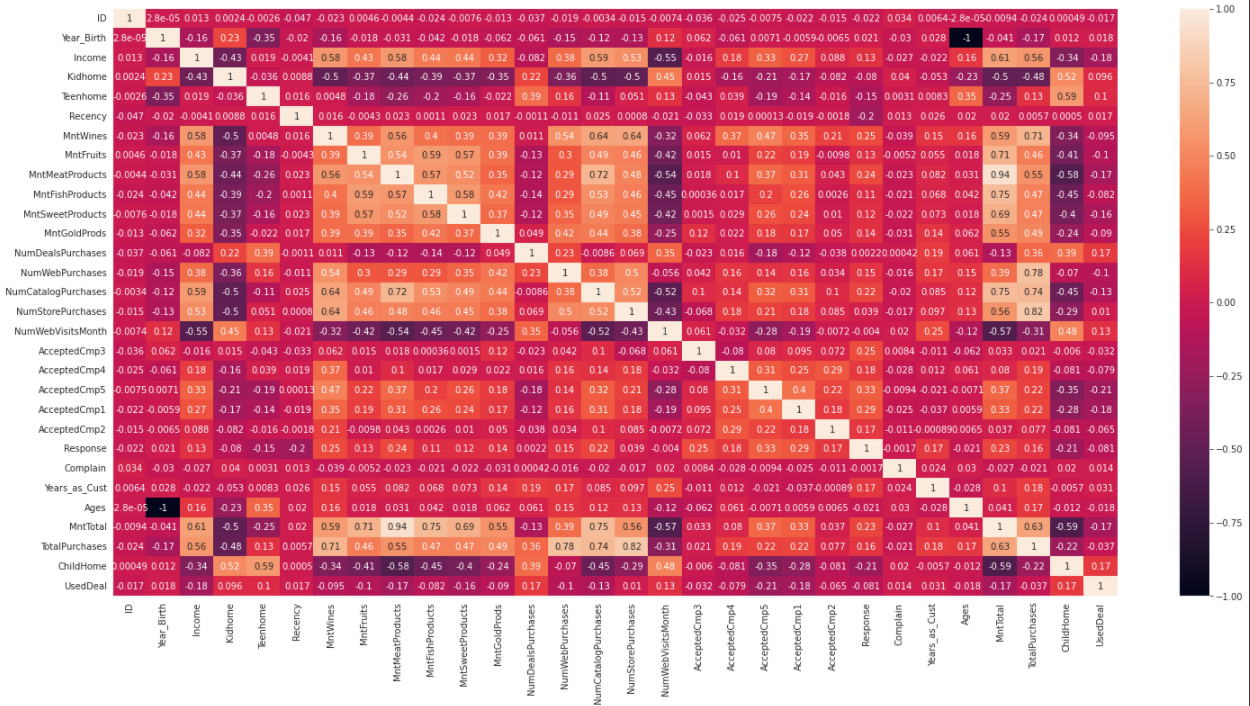
# Look for any correlation

plt.figure(figsize=(25,12))

sns.heatmap(mktdata\_E.corr(), annot=True)

plt.show()

Output:



Code 11: Correlation between variables

I could see the “Response” is negatively correlated with some variables such as: “Recency”, and “ChildHome” while it has positive correlations with the “MntWines”, “MntMeatProducts”, “MntTotal”, “NumCatalogPurchases” and all the acceptance toward the 5 last marketing campaigns.

# MODELLING

I tried to address 2 main questions with this modelling analysis:

* Are there any patterns in customers’ profiles, purchasing channels or purchased products that could be used to identify the key targets to focus on in the next campaigns for the improvement of acceptance rates? I have tried to apply clustering analysis for this part and added new indicator variables which tided to each cluster into the original dataset and applied further modelling analysis
* And could I predict the success rate of the next marketing campaign using the historical data including the findings from cluster analysis? Since the target is binary so I have applied Decision Tree, Logistic Regression, Gradient Boosting Classifier and Neural Network to predict the target. I have also applied GidSearchCV for tunning the hyperparameter to get the optimal model.

## Cluster Analysis

from sklearn.cluster import KMeans

kmeans\_models = [KMeans(n\_clusters=k, random\_state=23).fit(X\_red) for k in range (1, 10)]

innertia = [model.inertia\_ for model in kmeans\_models]

plt.plot(range(1, 10), innertia)

plt.title('Elbow method')

plt.xlabel('Number of Clusters')

plt.ylabel('WCSS')

plt.show()

from sklearn.metrics import silhouette\_score

silhoutte\_scores = [silhouette\_score(X\_red, model.labels\_) for model in kmeans\_models[1:4]]

plt.plot(range(2,5), silhoutte\_scores, "bo-")

plt.xticks([2, 3, 4, 5])

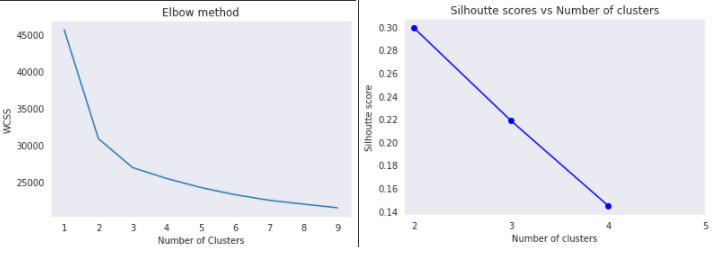
plt.title('Silhoutte scores vs Number of clusters')

plt.xlabel('Number of clusters')

plt.ylabel('Silhoutte score')

plt.show()

Output



The Elbow method is used to find the optimal number of clusters. Here we can see the WCSS score which is the sum of square distances from each point to its assigned center was reduced by the number of clusters increasing. With k > 3, the score curve becomes flattened and reaches near to X-axis, which means the square distances become smaller and smaller and each cluster is not significantly separated.

Another method to optimize k is using Silhouette coefficients, if the Silhouette score is near +1, it indicates that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster. In our case, for k = 4,5 the score is too small which means the clusters are not separated so k = 3 should be the optimal number of clusters.

I then need to add the cluster names and indicators for each observation and continue with the analysis to see if there are any differences between clusters in terms of customers’ profiles, purchase behaviors …

#adding cluster labels column into dataset

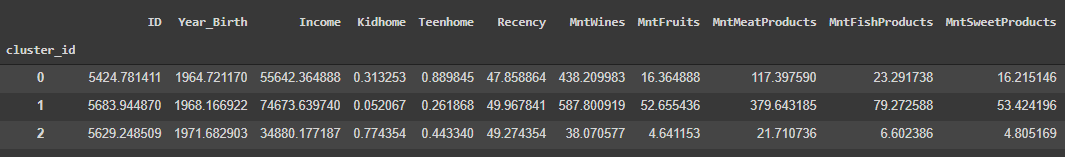
mktdata\_E['cluster\_id'] = kmeans.labels\_

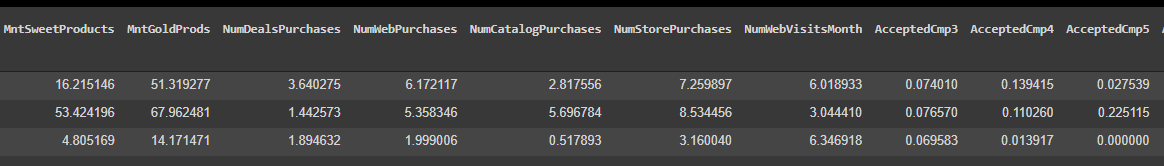
mktdata\_trf['cluster\_id'] = kmeans.labels\_

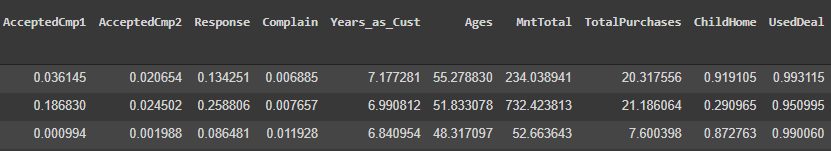
mktdata\_trf = pd.get\_dummies(mktdata\_trf, columns=['cluster\_id'])

mktdata\_E.groupby('cluster\_id').mean()

Output







I saw the differences between clusters:

* Cluster 1’s customers have the highest income and Cluster 2’s customers have the lowest income
* Cluster 1’s customers are more likely to have a child in their household
* Cluster 1’s customers tend to buy more stuff than others and they don’t care much about deals and do not often visit web

Based on the demographic of each group, I tried to label them as their characteristics

Cluster\_id = 2: Low income & less spending = L&L

Cluster\_id = 1: High income & Loyal = H&H

Cluster\_id = 0: Medium income & medium spending = M&M

Next before applying any predictive models, I applied dataset partition

# Partition and transform data before modeling

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import confusion\_matrix, accuracy\_score

# isolate X and y variables, and perform train-test split

X = mktdata\_trf.drop(['Response'], axis=1)

y = mktdata\_trf['Response']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=1)

Code 12: Data partition

## Logistic Regression

Apply Logistic Regression to predict the response output

# Logistic regression model

lored = LogisticRegression()

lored.fit(X\_train, y\_train)

# predictions

y\_pred = lored.predict(X\_test)

# evaluate model using RMSE

print("Logistic regression model RMSE: ", np.sqrt(mean\_squared\_error(y\_test, y\_pred)))

print("Median value of target variable: ", y.median())

# import the metrics class

from sklearn import metrics

cm = metrics.confusion\_matrix(y\_test, y\_pred)

cls\_names=[0,1] # name  of classes

fig, ax = plt.subplots()

ticks = np.arange(len(cls\_names))

plt.xticks(ticks, cls\_names)

plt.yticks(ticks, cls\_names)

# create heatmap

sns.heatmap(pd.DataFrame(cm), annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set\_label\_position("top")

plt.tight\_layout()

plt.title('Confusion matrix\n', fontsize = 16)

plt.ylabel('Actual label',fontsize = 16)

plt.xlabel('Predicted label',fontsize = 16)

# Print accuracy, Precision, Recall and F1 Score

from sklearn.metrics import precision\_score, recall\_score, f1\_score

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

print("Precision:",metrics.precision\_score(y\_test, y\_pred, average="binary"))

print("Recall:",metrics.recall\_score(y\_test, y\_pred, average="binary"))

print("F1 Score:",metrics.f1\_score(y\_test, y\_pred, average="binary"))

#Create DataFrame of coefficient

coefficient = pd.DataFrame({'Feature': X\_train.columns, 'coefficient' : lored.coef\_[0]})

coefficient = coefficient.sort\_values(by=['coefficient'],ascending=False)

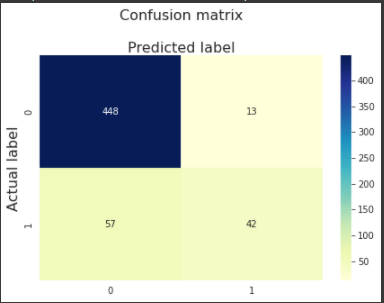
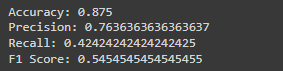
# plot the coefficient

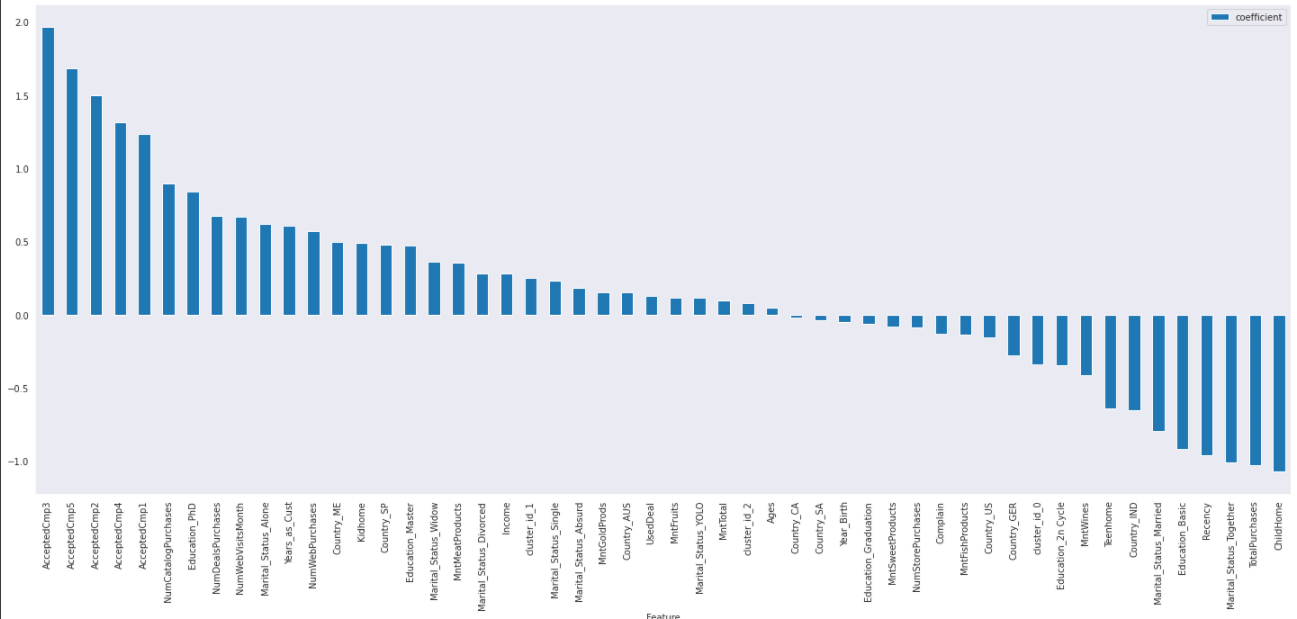
ax = coefficient.plot.bar(x='Feature', y='coefficient', figsize=(25,10),rot=0)

plt.xticks(rotation=90)

Output

Logistic regression model RMSE: 0.3535533905932738





Code 13: Apply Logistic Regression

## Decision Tree

from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier

from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

from sklearn.datasets import load\_iris

from sklearn.model\_selection import cross\_val\_score

# Create Decision Tree classifer object

clf = DecisionTreeClassifier()

# Train Decision Tree Classifer

clf = clf.fit(X\_train,y\_train)

#Predict the response for test dataset

y\_pred = clf.predict(X\_test)

# evaluate model using RMSE

print("Decision Tree model RMSE: ", np.sqrt(mean\_squared\_error(y\_test, y\_pred)))

print("Median value of target variable: ", y.median())

print("Max depth= " + str(clf.get\_depth()))

print("Number of Leaves= " + str(clf.get\_n\_leaves()))

importance = pd.DataFrame({'Feature': X\_train.columns, 'Importance Score' : clf.feature\_importances\_})

importance = importance.sort\_values(by=['Importance Score'],ascending=False)

# plot feature importance

ax = importance.plot.bar(x='Feature', y='Importance Score', figsize=(25,10),rot=0)

plt.xticks(rotation=90)

cls\_names=[0,1] # name  of classes

fig, ax = plt.subplots()

ticks = np.arange(len(cls\_names))

plt.xticks(ticks, cls\_names)

plt.yticks(ticks, cls\_names)

# create heatmap

sns.heatmap(pd.DataFrame(cm), annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set\_label\_position("top")

plt.tight\_layout()

plt.title('Confusion matrix\n', fontsize = 16)

plt.ylabel('Actual label',fontsize = 16)

plt.xlabel('Predicted label',fontsize = 16)

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

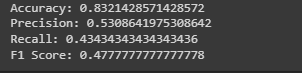
print("Precision:",metrics.precision\_score(y\_test, y\_pred, average="binary"))

print("Recall:",metrics.recall\_score(y\_test, y\_pred, average="binary"))

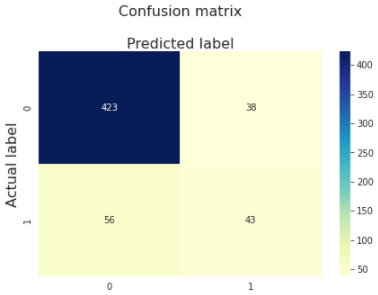
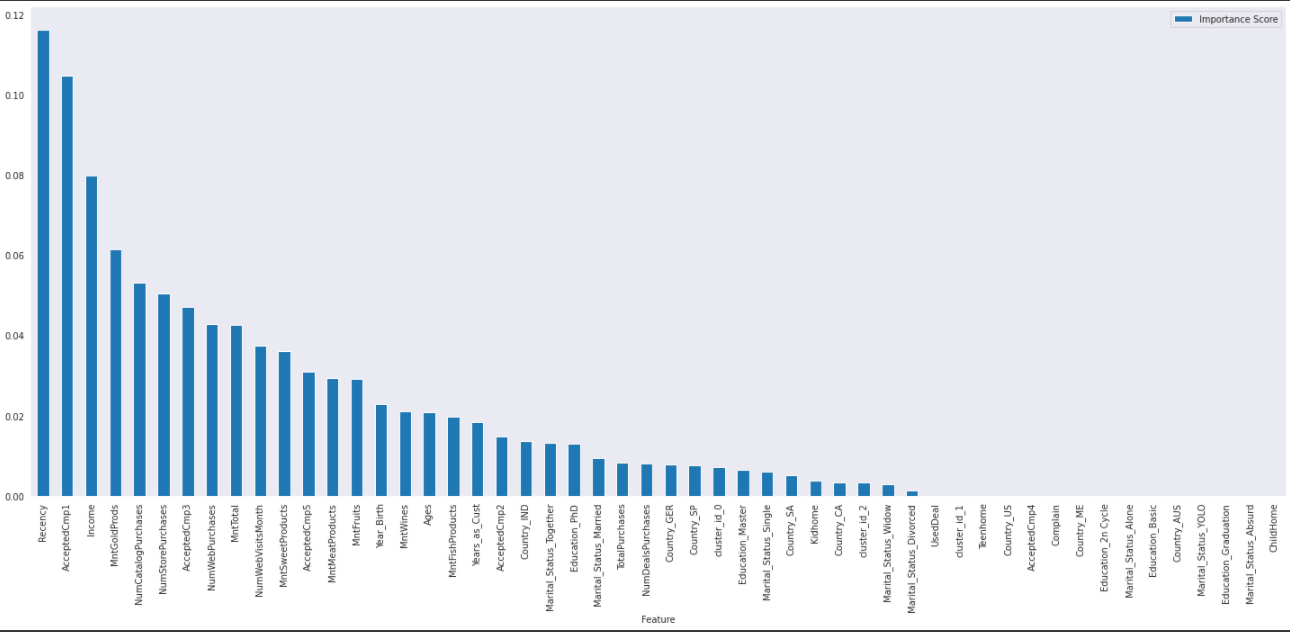
print("F1 Score:",metrics.f1\_score(y\_test, y\_pred, average="binary"))

Output:

Decision Tree model RMSE: 0.4097037257057139





2 

Code 14: Apply Maximal Decision Tree

The maximal tree model has a higher RMSE and lower accuracy than the Logistic Regression model. Based on the results of features’ importance levels, I could see some variables are contributing much in predicting the target are: “Recency”, “AcceptedCmp1”, “Income”, “MntGoldProds”…

I have tried to tune the model to get the optimal model using GridsearchCV

from sklearn.model\_selection import GridSearchCV

param\_dict = {"criterion": ['gini', 'entropy' ],

              "max\_depth": range (1, 20),

              "min\_samples\_split": range (1, 10),

              "min\_samples\_leaf": range (1, 5)}

grid = GridSearchCV(clf,

                    param\_grid = param\_dict,

                    cv=10,

                    verbose=1,

                    n\_jobs=-1)

grid.fit(X\_train, y\_train)

print(str(grid.best\_estimator\_))

print(grid.best\_score\_)

Output:



After I got the optimal hyperparameters, I then applied the Optimal Decision Tree model

# Create Decision Tree classifer object

clf\_op = grid.best\_estimator\_

# Train Decision Tree Classifer

clf\_op = clf\_op.fit(X\_train,y\_train)

#Predict the response for test dataset

y\_pred = clf\_op.predict(X\_test)

# evaluate model using RMSE

print("Optimal Decision Tree model RMSE: ", np.sqrt(mean\_squared\_error(y\_test, y\_pred)))

print("Median value of target variable: ", y.median())

print("Max depth= " + str(clf\_op.get\_depth()))

print("Number of Leaves= " + str(clf\_op.get\_n\_leaves()))

importance = pd.DataFrame({'Feature': X\_train.columns, 'Importance Score' : clf\_op.feature\_importances\_})

importance = importance.sort\_values(by=['Importance Score'],ascending=False)

# plot feature importance

ax = importance.plot.bar(x='Feature', y='Importance Score', figsize=(25,10),rot=0)

plt.xticks(rotation=90)

cm = metrics.confusion\_matrix(y\_test, y\_pred)

cls\_names=[0,1] # name  of classes

fig, ax = plt.subplots()

ticks = np.arange(len(cls\_names))

plt.xticks(ticks, cls\_names)

plt.yticks(ticks, cls\_names)

# create heatmap

sns.heatmap(pd.DataFrame(cm), annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set\_label\_position("top")

plt.tight\_layout()

plt.title('Confusion matrix\n', fontsize = 16)

plt.ylabel('Actual label',fontsize = 16)

plt.xlabel('Predicted label',fontsize = 16)

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

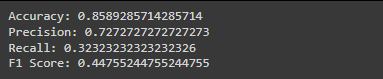
print("Precision:",metrics.precision\_score(y\_test, y\_pred, average="binary"))

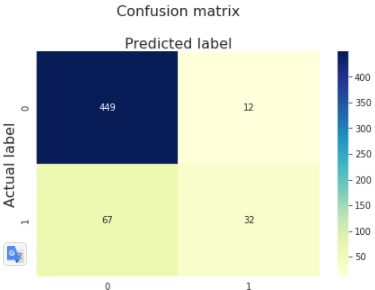
print("Recall:",metrics.recall\_score(y\_test, y\_pred, average="binary"))

print("F1 Score:",metrics.f1\_score(y\_test, y\_pred, average="binary"))

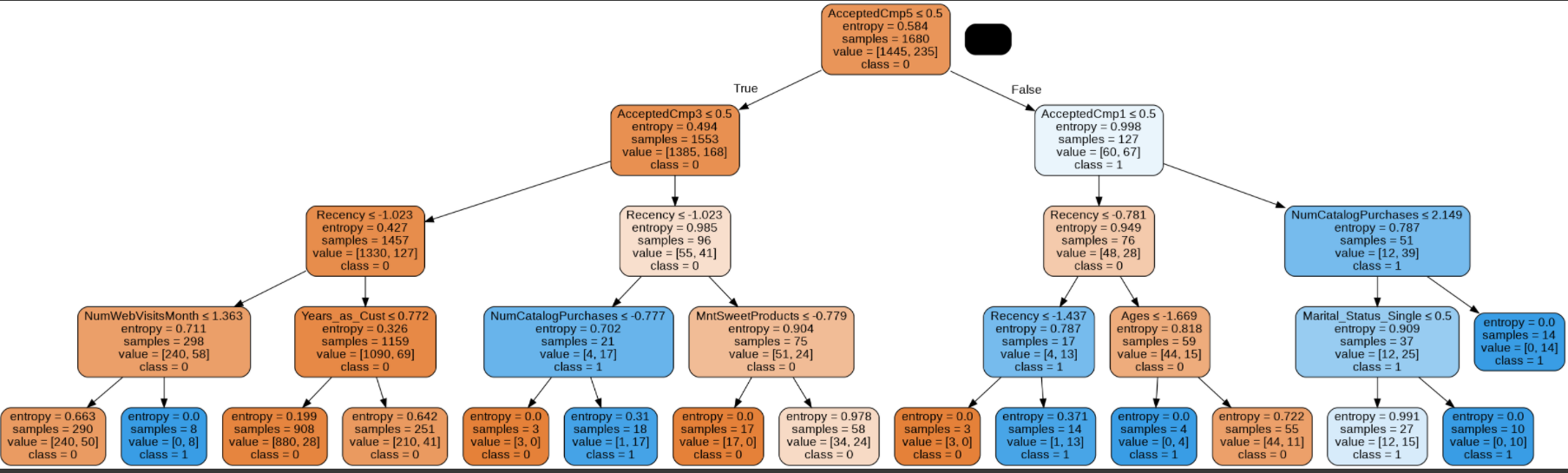
Output:

Optimal Decision Tree model RMSE: 0.3755947664324259









Code 15: Apply optimal Decision Tree

The optimal tree model has been improved but it still has a higher RMSE and lower accuracy than the Logistic Regression model. Based on the results of features’ importance levels, I could see some variables are contributing much in predicting the target are: “AcceptedCmp5”, “Recency”, “AcceptedCmp3”, “Year\_as\_Cust”…

## Gradient Boosting Classifier

from sklearn.ensemble import GradientBoostingClassifier

classifier\_GBC = GradientBoostingClassifier(random\_state = 3)

classifier\_GBC.fit(X\_train, y\_train)

y\_pred = classifier\_GBC.predict(X\_test)

# evaluate model using RMSE

print("Gradient Booster Classifier model RMSE: ", np.sqrt(mean\_squared\_error(y\_test, y\_pred)))

# Model Accuracy, how often is the classifier correct?

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

print("Precision:",metrics.precision\_score(y\_test, y\_pred, average="binary"))

print("Recall:",metrics.recall\_score(y\_test, y\_pred, average="binary"))

print("F1 Score:",metrics.f1\_score(y\_test, y\_pred, average="binary"))

Output:

Gradient Booster Classifier model RMSE: 0.3635145899999849

Accuracy: 0.8678571428571429

Precision: 0.7906976744186046

Recall: 0.3434343434343434

F1 Score: 0.47887323943661964

The accuracy was still not good enough so I applied Randomized Search to tune the model

from sklearn.model\_selection import RandomizedSearchCV

grid = {

    'learning\_rate' : [0.2, 0.3, 0.4, 0.5],

    'n\_estimators' : [300, 500, 700, 900],

    'min\_samples\_split' : [3, 4, 5, 6],

    'max\_depth' : [2, 3, 4, 5],

    'loss' : ['deviance', 'exponential']

}

random\_cv = RandomizedSearchCV(estimator=classifier\_GBC,

                              param\_distributions=grid,

                              n\_iter=20,

                              n\_jobs=-1,

                              cv=5,

                              verbose=7,

                              random\_state=10,

                              scoring='accuracy')

random\_cv.fit(X\_train, y\_train)

random\_cv.best\_estimator\_

Output:  


Apply the optimal model to the dataset

hgb = random\_cv.best\_estimator\_

hgb.fit(X\_train, y\_train)

y\_pred = hgb.predict(X\_test)

# evaluate model using RMSE

print("Gradient Booster Classifier model RMSE: ", np.sqrt(mean\_squared\_error(y\_test, y\_pred)))

print("Median value of target variable: ", y.median())

cls\_names=[0,1] # name  of classes

fig, ax = plt.subplots()

ticks = np.arange(len(cls\_names))

plt.xticks(ticks, cls\_names)

plt.yticks(ticks, cls\_names)

# create heatmap

sns.heatmap(pd.DataFrame(cm), annot=True, cmap="YlGnBu" ,fmt='g')

ax.xaxis.set\_label\_position("top")

plt.tight\_layout()

plt.title('Confusion matrix\n', fontsize = 16)

plt.ylabel('Actual label',fontsize = 16)

plt.xlabel('Predicted label',fontsize = 16)

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

print("Precision:",metrics.precision\_score(y\_test, y\_pred, average="binary"))

print("Recall:",metrics.recall\_score(y\_test, y\_pred, average="binary"))

print("F1 Score:",metrics.f1\_score(y\_test, y\_pred, average="binary"))

importance = pd.DataFrame({'Feature': X\_train.columns, 'Importance Score' : hgb.feature\_importances\_})

importance = importance.sort\_values(by=['Importance Score'],ascending=False)

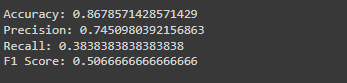
# plot feature importance

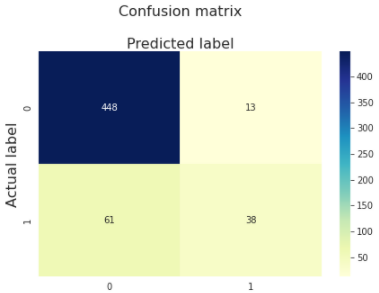
ax = importance.plot.bar(x='Feature', y='Importance Score', figsize=(25,10),rot=0)

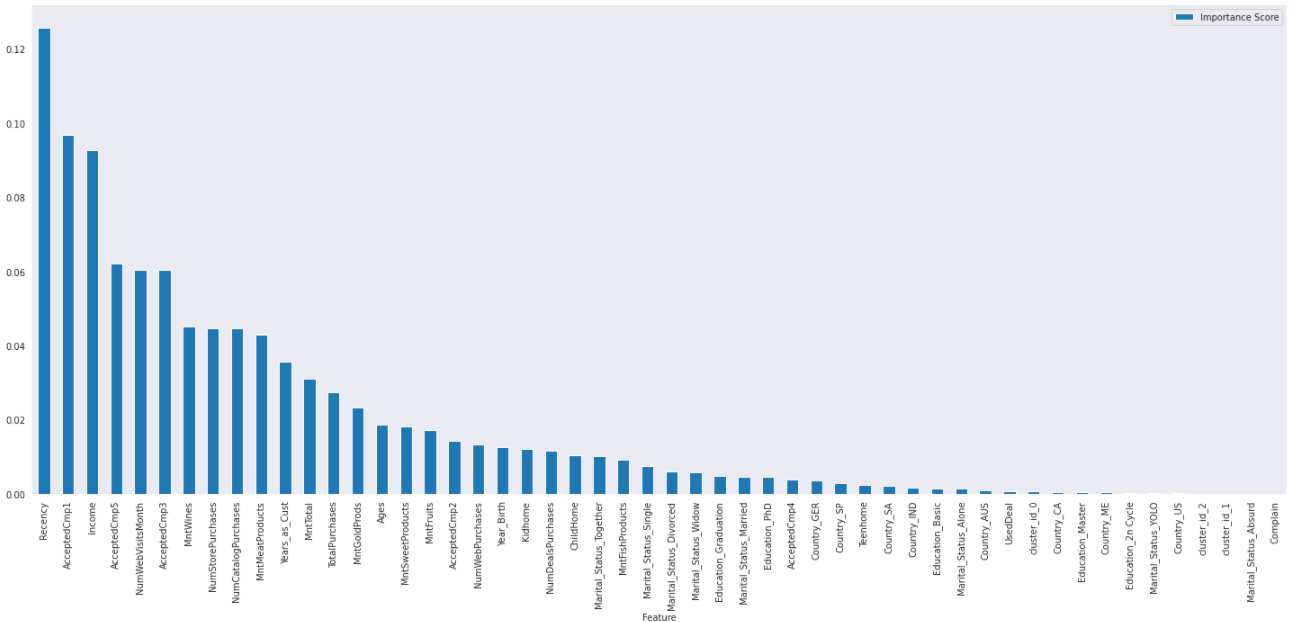
plt.xticks(rotation=90)

Output:

Gradient Booster Classifier model RMSE: 0.35606981658898623







Code 16: Apply Gradient Boosting Classifier

I could see the improvement in terms of RMSE and accuracy but again it was not good as the Logistic Regression model.

## Neural Network

from keras.models import Sequential

from keras.layers import Dense, Flatten

from keras import optimizers

# set Hyper parameters

learning\_rate=0.01

no\_epochs=100

# Model creation

model = Sequential([

    Flatten(input\_shape=(52,)),

    Dense(16, activation='relu'),

    Dense(16, activation='relu'),

    Dense(1, activation='sigmoid'),

])

#Compile model

model.compile(optimizer='adam',

              loss='binary\_crossentropy',

              metrics=['accuracy'])

# Fit model

model.fit(X\_train, y\_train, epochs=no\_epochs, batch\_size=len(mktdata\_trf),  verbose=2)

y\_pred = model.predict(X\_test)

test\_loss, test\_acc = model.evaluate(X\_test, y\_test)

print('Test accuracy:', test\_acc)

Output:

Test accuracy: 0.8410714268684387

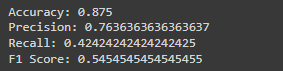
Code 17: Apply Neural Network

The Neural Network model seems to have the lowest accuracy level.

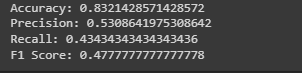
## Model Comparison

As the final results show the Logistic Regression model has the smallest RMSE and highest Accuracy and F1 Score. I have chosen Logistic Regression as the model to go with and predict the target variable which means the acceptance toward the next campaign.

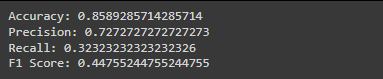
Logistic regression model RMSE: 0.3535533905932738



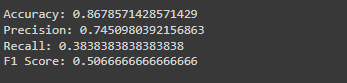
Decision Tree model RMSE: 0.4097037257057139



Optimal Decision Tree model RMSE: 0.3755947664324259



Gradient Booster Classifier model RMSE: 0.35606981658898623



Neural Network accuracy: 0.8410714268684387

## Find the Significant Features when applying Logistic Regression

Next, I tried to see which variables are significant in predicting the target by showing the P-values together with the summary of the model. Then I tried to eliminate the less meaningful or less important features in the model to reduce the complexity.

Firstly, I rerun the Logistic Regression with “statsmodels api” and tried to show the summary. The error occurred “LinAlgError: Singular matrix”, this should be because of highly correlated variables in the input data. Fitting the data with high correlates into the maximum likelihood like in “Statsmodel” is going to be highly unstable

import statsmodels.api as sm

log\_reg = sm.Logit(y\_train, X\_train).fit()

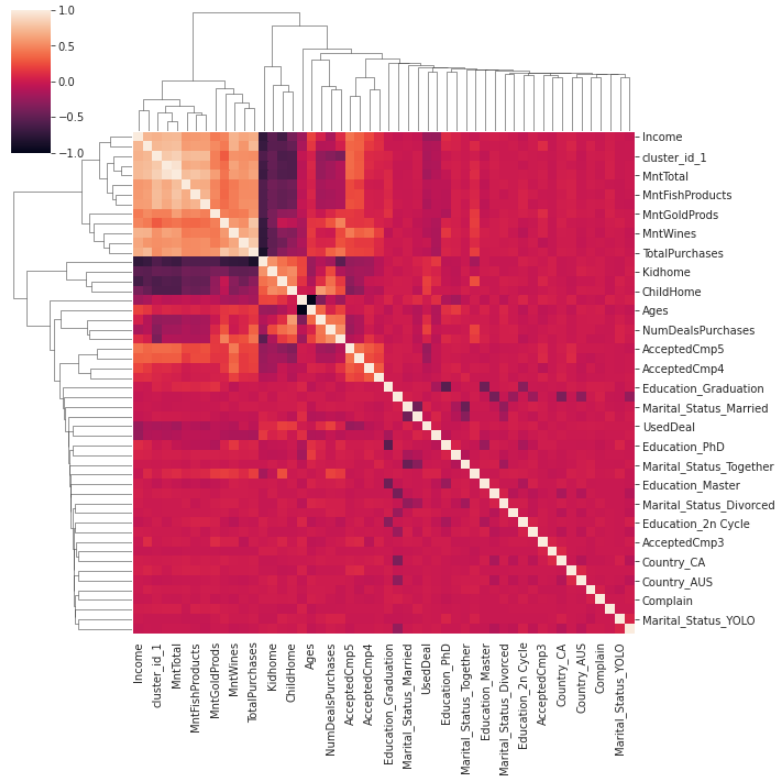
print(log\_reg.summary())

Output:

LinAlgError: Singular matrix

sns.clustermap(X\_train.corr())

Output:



I could see some features that are highly correlated to each other, such as:

“ Income”, “MntTotal”…

Next, I tried to remove some of the non-significant variables or less important variables in the input data. In that way, I could reduce the complexity and also could apply the “statsmodels”.

RFE or Recursive Feature Elimination is the features selection method that assigns weights to features (e.g., the coefficients of a linear model), the goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. It begins with the full set of features and goes backward until reaching the specific number of features.

I have applied RFE several times and reapplied to the Logit Regression from “Statsmodels” until I got the number of features equal to 25 which can fit the model:

rfe = RFE(lored,n\_features\_to\_select=25)

rfe = rfe.fit(X\_train, y\_train.values)

print(rfe.support\_)

print(rfe.ranking\_)

Output:

[False False True False False True False False False True True True

False False False False True True True True True True True True

True True True False False True False False True True True False

True False True False False False False True True True False False

False False False False]

[27 11 1 7 21 1 19 24 16 1 1 1 12 2 22 23 1 1 1 1 1 1 1 1

1 1 1 14 10 1 13 6 1 1 1 17 1 15 1 26 25 9 5 1 1 1 4 28

3 8 20 18]

Then I create the new input data with 25 features and called it “X\_train\_sig”:

X\_train\_sig = X\_train.loc[:, [False, False, True, False, False, True , False, False, False, True, True, True

                              , False, True, False, False, True, True, True, True, True, True, True, True

                              , True, False, True, False, False, True, False, True, True, False, True , False

                              , True, False, True, False, False, False, False, False, True, False, False, True

                              , False, True, False, False]]

X\_test\_sig = X\_test.loc[:, [False, False, True, False, False, True , False, False, False, True, True, True

                              , False, True, False, False, True, True, True, True, True, True, True, True

                              , True, False, True, False, False, True, False, True, True, False, True , False

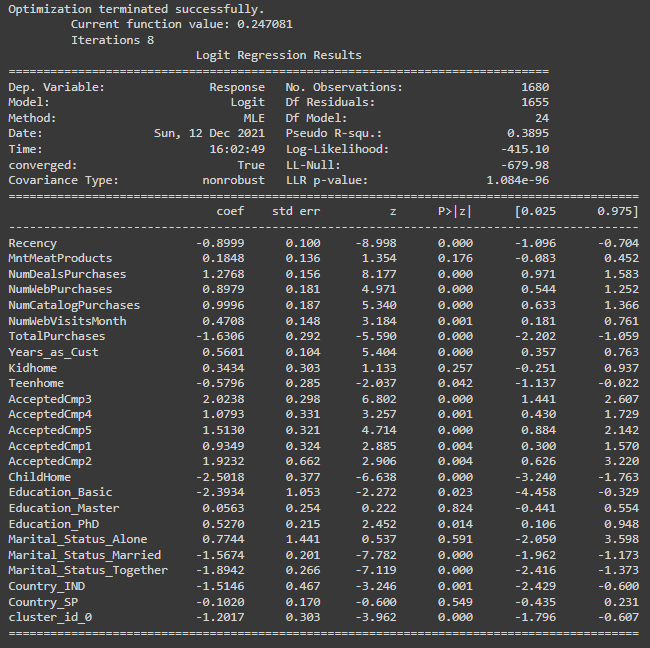
                              , True, False, True, False, False, False, False, False, True, False, False, True

                              , False, True, False, False]]

log\_reg = sm.Logit(y\_train, X\_train\_sig).fit()

print(log\_reg.summary())

Output:



y\_pred = log\_reg.predict(X\_test\_sig)

# change the y\_pred from probability to decrete value 0 and 1

for idx in y\_pred.index :

  if y\_pred[idx] < 0.5:

    y\_pred[idx] = 0

  else:

    y\_pred[idx] = 1

print("Logistic regression with 24 features model RMSE: ", np.sqrt(mean\_squared\_error(y\_test, y\_pred)))

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

print("Precision:",metrics.precision\_score(y\_test, y\_pred, average="binary"))

print("Recall:",metrics.recall\_score(y\_test, y\_pred, average="binary"))

print("F1 Score:",metrics.f1\_score(y\_test, y\_pred, average="binary"))

Output:

Logistic regression with 24 features model RMSE: 0.34589428608009287

Accuracy: 0.8803571428571428

Precision: 0.7758620689655172

Recall: 0.45454545454545453

F1 Score: 0.5732484076433122

Code 18: Applying RFE to find the significant variables

As the result showed, I had 19 variables that have the P-values that < 0.05 and I can reject the Null Hypothesis and conclude that these 19 variables are significant in predicting the target acceptance toward the next marketing campaign.

When running the model to predict the test set, I have got the highest scores so far with the accuracy = 0.88 and F1-score = 0.57. This means the model was getting better in predicting the target.

# CONCLUSION

Next step, I created a data input with all the observations and 25 features and added the predicted target into the dataset and visualize the predicted output by groups.

# Create dataframe mktdata\_trf\_sig to predict target with 24 features

mktdata\_trf\_sig =mktdata\_trf.drop(['Response'], axis=1).loc[:, [False, False, True, False, False, True , False, False, False, True, True, True

                              , False, True, False, False, True, True, True, True, True, True, True, True

                              , True, False, True, False, False, True, False, True, True, False, True , False

                              , True, False, True, False, False, False, False, False, True, False, False, True

                              , False, True, False, False]]

y\_pred\_sig = log\_reg.predict(mktdata\_trf\_sig)

# change the y\_pred from probability to decrete value 0 and 1

for idx in y\_pred\_sig.index :

  if y\_pred\_sig[idx] < 0.5:

    y\_pred\_sig[idx] = 0

  else:

    y\_pred\_sig[idx] = 1

mktdata\_E['Response\_pred'] = y\_pred\_sig

CrosstabResult = pd.crosstab(index=mktdata\_E['Response\_pred'],columns=mktdata\_E['cluster\_cat'])

CrosstabResult.loc['%Repsonse'] = (CrosstabResult.loc[1]/ (CrosstabResult.loc[1]+CrosstabResult.loc[0]))\*100

print(CrosstabResult)

CrosstabResult.plot.bar(figsize=(7,4), rot=0)

Output:



Code : Response numbers and rates by cluster

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | coef | std err | z | P>|z| |
| Recency | -0.8999 | 0.1 | -8.998 | 0.000 |
| NumDealsPurchases | 1.2768 | 0.156 | 8.177 | 0.000 |
| NumWebPurchases | 0.8979 | 0.181 | 4.971 | 0.000 |
| NumCatalogPurchases | 0.9996 | 0.187 | 5.34 | 0.000 |
| TotalPurchases | -1.6306 | 0.292 | -5.59 | 0.000 |
| Years\_as\_Cust | 0.5601 | 0.104 | 5.404 | 0.000 |
| AcceptedCmp3 | 2.0238 | 0.298 | 6.802 | 0.000 |
| AcceptedCmp5 | 1.513 | 0.321 | 4.714 | 0.000 |
| ChildHome | -2.5018 | 0.377 | -6.638 | 0.000 |
| Marital\_Status\_Married | -1.5674 | 0.201 | -7.782 | 0.000 |
| Marital\_Status\_Together | -1.8942 | 0.266 | -7.119 | 0.000 |
| cluster\_id\_0 | -1.2017 | 0.303 | -3.962 | 0.000 |
| NumWebVisitsMonth | 0.4708 | 0.148 | 3.184 | 0.001 |
| AcceptedCmp4 | 1.0793 | 0.331 | 3.257 | 0.001 |
| Country\_IND | -1.5146 | 0.467 | -3.246 | 0.001 |
| AcceptedCmp1 | 0.9349 | 0.324 | 2.885 | 0.004 |
| AcceptedCmp2 | 1.9232 | 0.662 | 2.906 | 0.004 |
| Education\_PhD | 0.527 | 0.215 | 2.452 | 0.014 |
| Education\_Basic | -2.3934 | 1.053 | -2.272 | 0.023 |
| Teenhome | -0.5796 | 0.285 | -2.037 | 0.042 |

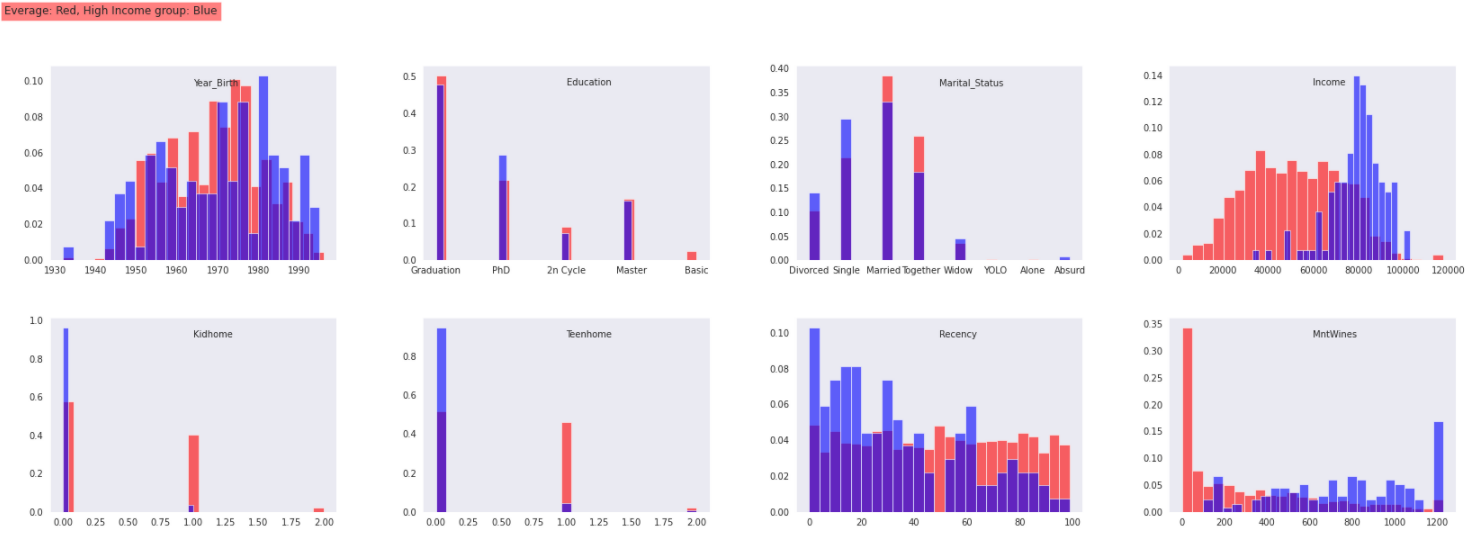
Table 2: Logistic Regression Result summary for 19 Features fitting

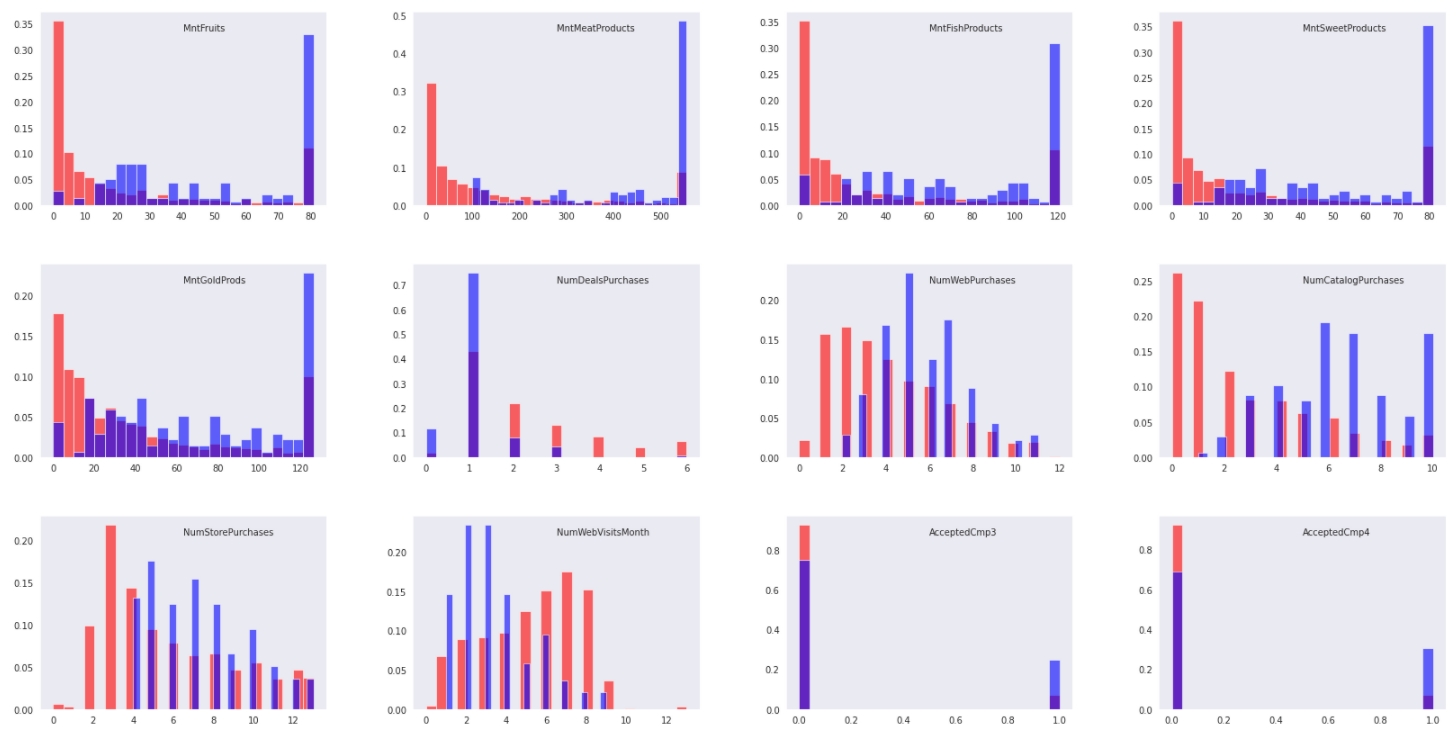
The result showed that:

* Cluster 1’s customers (or H&H group) are most likely to respond to the new campaign based on the prediction, while group L&L is less likely to respond
* Cluster\_id\_0 is significant and has a big negative coefficient which means we should not target this group in the next campaign since they are most likely to say no and they significantly and negatively affected the total acceptance rate.
* We also should not focus on the having-child, married or stay together household since they all significantly and negatively affect the total acceptance rate
* It seems like the higher education the higher rate of responses so we should focus more on the higher education group
* We should also focus on the customers that previously say “yes” to campaigns 3 and 5 and of course, the more loyal they are, the more possibility they would say “yes” to our next campaign.

Below are some of the visualizations that demonstrated the differences between groups in terms of profiles and purchasing behaviors in the last 2 years.

Group H&H:





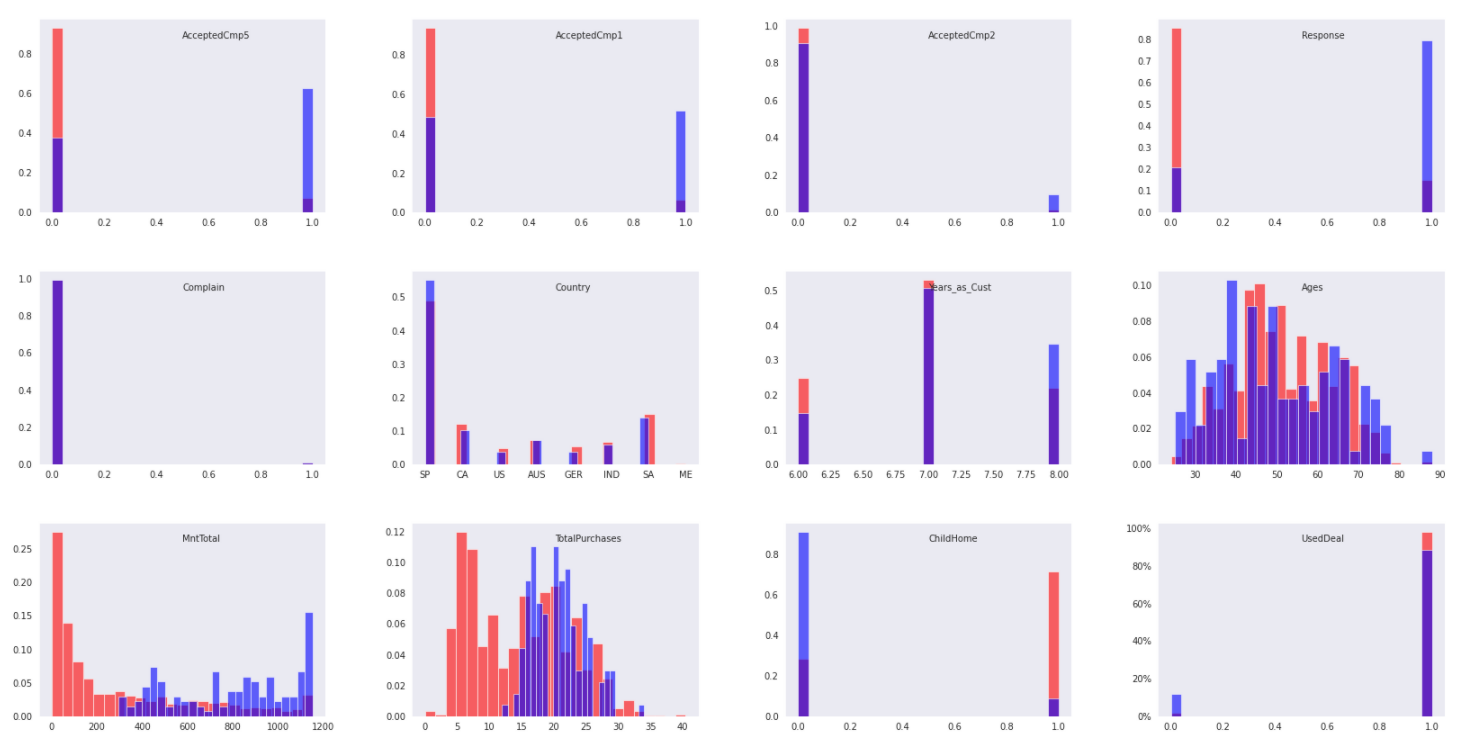
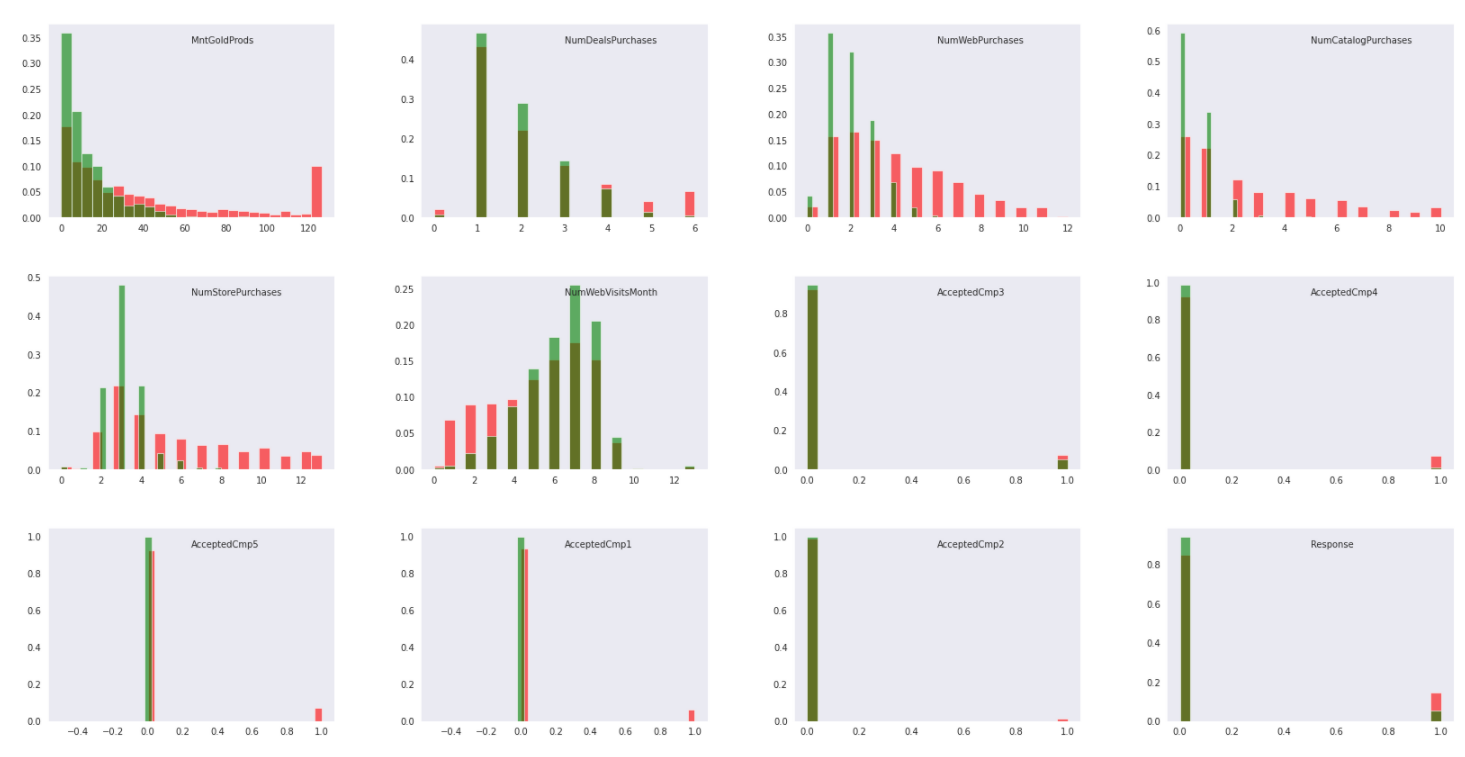


Figure 5: Profiles and behaviors of H&H group vs average

Group L&L





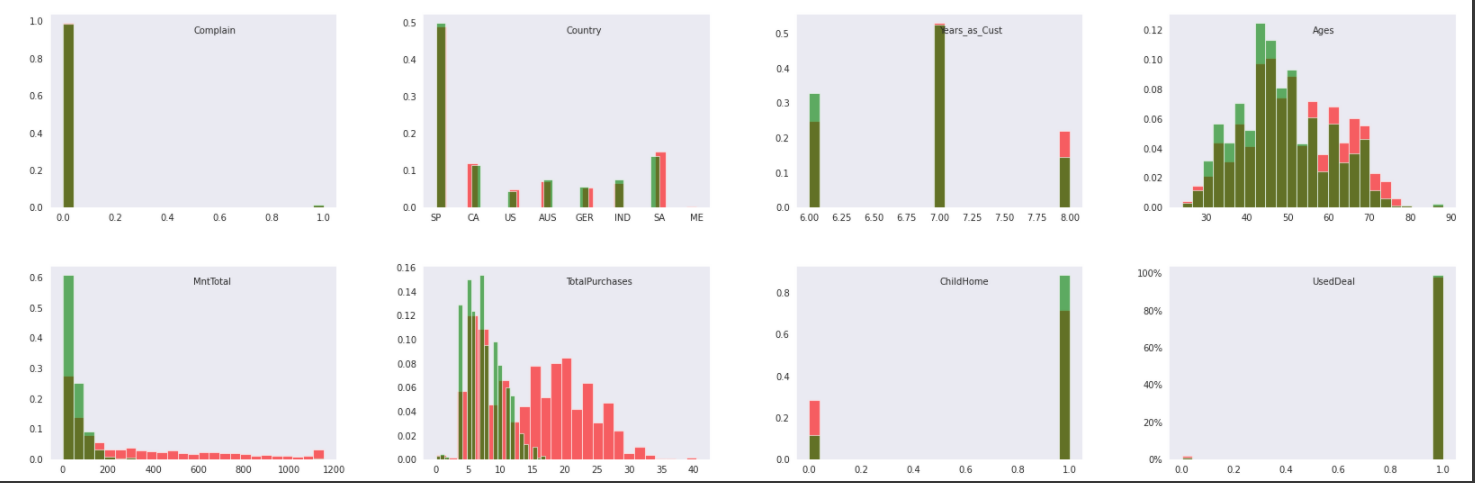


Figure 6: Profiles and behaviors of L&L group vs average

Group M&M



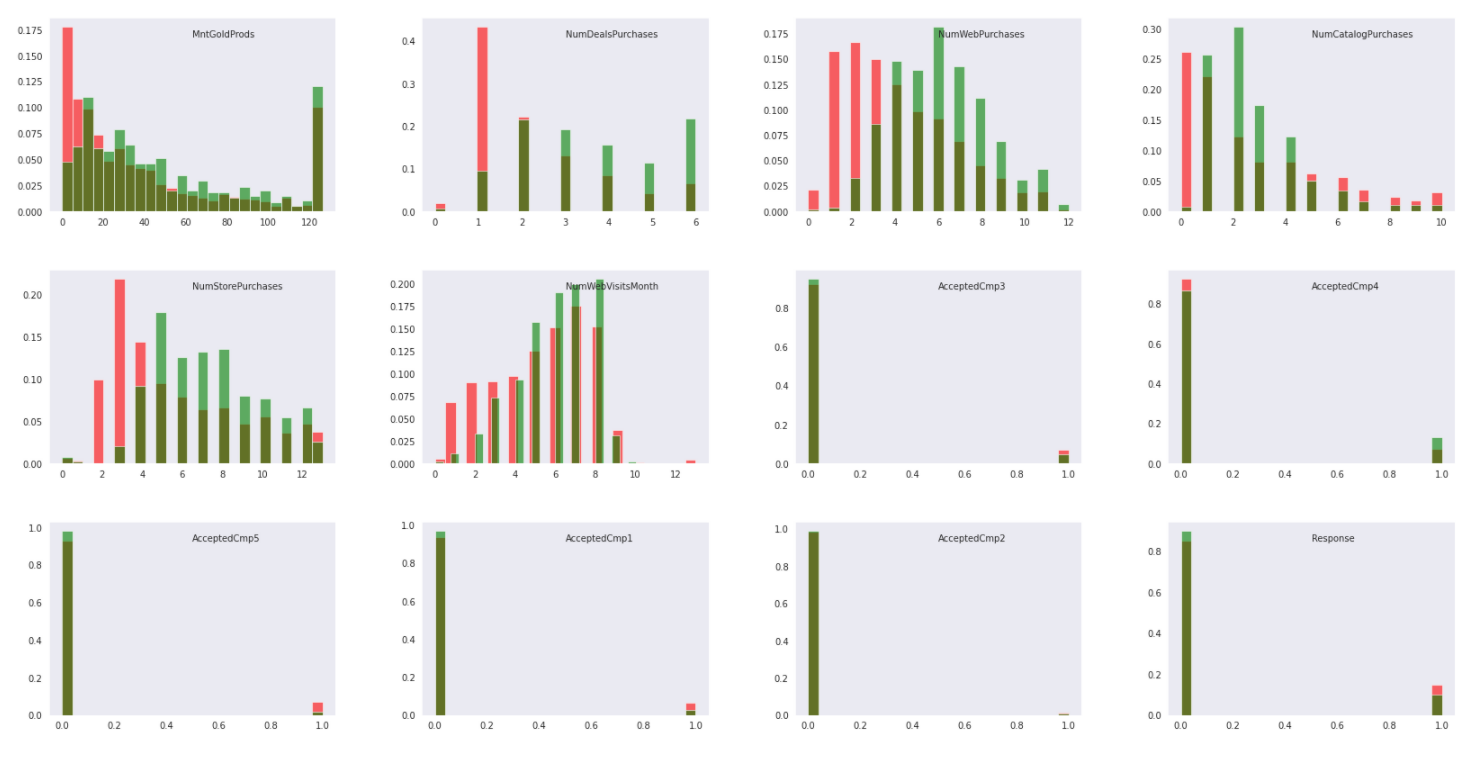


Figure 7: Profiles and behaviors of M&M group vs average

# GOVERNANCE PLAN

## Risks Management

When applying models and algorithms in machine learning, if we do not have the appropriate governance measures, risks will arise. The idea of this part is to assume and try to foresee what risks can affect the results of the model and also the decision-making process. The risks could potentially come from 3 main sources: input data, algorithm design and output data. As we are dealing with the sale dataset with limited information about customers’ profiles so these risks could be classified as Minimal Risk.

### Input

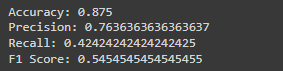
Let’s take a look back at the source of our dataset and how the data was collected and stored. The data was extracted from sale data of the period from December 2018 to December 2020 and it contained the profiles of our customers, their historical purchasing behaviors and their acceptance toward our marketing campaigns in 2020. Some risks could happen:

* Since we are predicting the acceptance rate of the next campaign so the data is outdated because it missed the sale data for the whole year 2021 so it could not reflect the current situation especially in the middle of the pandemic when people are changing their shopping behaviors. The result could be misleading and it is risky to apply it to the next marketing campaign.
* This dataset missed an important part which is the information about our customers’ purchasing behaviors within each marketing campaign and what did we offer in each marketing campaign. It could be highly correlated to the target of acceptance toward the new marketing campaign
* The profile information was collected mostly from 2014 and some features should have been changed such as Marital\_Status or ChildHome… There is no clue that these features have been updated and the time of the updates was not recorded. It also could change the result of our model since the values of some features have changed but have not been recorded
* We need to update the data regularly or in real-time and retrain all the models for better precision and choose the right model for a better prediction.

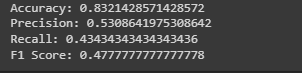
### . Algorithms

As shown in the conclusion part, the Root Mean Squared Error (RMSE) of the chosen model is 0.35 which is the lowest RMSE in all of the used models. 0.35 is the average distance between the actual acceptance in the test dataset and the predicted values. The model has the highest accuracy and F1-Score. I listed below the scores of each model when applied to predict the test dataset:

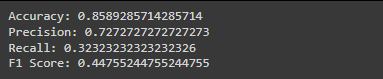
Logistic regression model RMSE: 0.3535533905932738



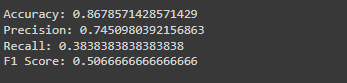
Decision Tree model RMSE: 0.4097037257057139



Optimal Decision Tree model RMSE: 0.3755947664324259

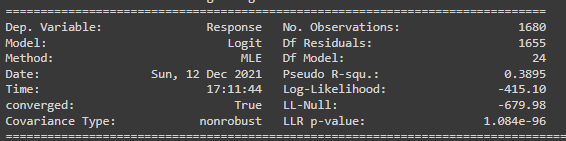


Gradient Booster Classifier model RMSE: 0.35606981658898623



Neural Network accuracy: 0.8410714268684387

I then based on the Logistic Regression model to find out which variables that are significant in predicting the target and retrain the model with just 24 significant features and got the better score as below:



Logistic regression with 24 features model RMSE: 0.34589428608009287

Accuracy: 0.8803571428571428

Precision: 0.7758620689655172

Recall: 0.45454545454545453

F1 Score: 0.5732484076433122

Code 20: Summary of Logistic Regression with 24 features

We can use this model to predict 38.95% of the variance of the target variable The LLR P-value of the model is also much smaller than 0.05 which demonstrates that we can reject the NULL hypothesis that using only the intercept to predict the target is better than using all the 24 variables. Applying the model to predict the test data we can see a pretty high accuracy of 88% and an F1 score of 57%. I could also simplify the model by removing unimportant variables and reducing the number of features to 24. This model is good with this dataset and the current input data to predict the next campaign. But for a better prediction, I recommend collecting more input data related to each marketing campaign and retraining all the models with the new input data. It would be time and cost-consuming because we would need to collect sufficient data to feed the model for retraining purposes. Meanwhile, for near-future campaign predictions:

* We can apply the Logistic Regression model with 24 features to predict the acceptance of the next campaign. We can calculate this RMSE with the upcoming dataset with this model and compare the 2 RMSEs to see if the model is still fit well with the future data or we need to retrain it with the new and more updated data input to get the better RMSE.
* If the RMSE rise above 0.3535 (or 2.3%) which is equal to the RMSE of the Logistic Regression model with 52 features, we need to run the Recursive Feature Elimination to find the best number of features to fit into the model
* If the RMSE rise above 0.356 (or 3.2%) which is equal to the RMSE of the Gradient Boosting Classifier model, we need to retrain all the models with new input data to find the new best fit model.
* As the result showed, we may not re-run the full Decision Tree model since it is quite complex and had the highest RMSE.

of input data but for long-term governance, we need to re-train the with updated data and choose the best fit model for prediction.

### . Output

As we are predicting the acceptance level toward a marketing campaign but the input data do not include much information about each marketing campaign and the next marketing campaign so the application of this output toward a completely strange campaign could lead to a wrong decision. For example, this model found that people who accepted campaign 3 and campaign 5 are more likely to accept this next campaign but if 2 campaigns are related to mostly Wine products but the next campaign only focuses on diapers which are irrelevant. The application of this model could lead to a low success rate. So there should be sufficient data input and the application of the output from our model need to be relevant to the input data.

## Variables level monitoring

Here are the accepted ranges of the data input variables. These ranges are determined based on the historical data and the forecast for the next casual coming future.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1. **Variables** | **mean** | **std** | **min** | **25%** | **50%** | **75%** | **max** |
| Year\_Birth | 1,968.81 | 11.98 | 1893 | 1959 | 1970 | 1977 | 1996 |
| Income | 52,237.98 | 25,037.96 | 1730 | 35538.75 | 51381.5 | 68289.75 | 666666 |
| Kidhome | 0.44 | 0.54 | 0 | 0 | 0 | 1 | 2 |
| Teenhome | 0.51 | 0.54 | 0 | 0 | 0 | 1 | 2 |
| Recency | 49.11 | 28.96 | 0 | 24 | 49 | 74 | 99 |
| MntWines | 303.94 | 336.60 | 0 | 23.75 | 173.5 | 504.25 | 1493 |
| MntFruits | 26.30 | 39.77 | 0 | 1 | 8 | 33 | 199 |
| MntMeatProducts | 166.95 | 225.72 | 0 | 16 | 67 | 232 | 1725 |
| MntFishProducts | 37.53 | 54.63 | 0 | 3 | 12 | 50 | 259 |
| MntSweetProducts | 27.06 | 41.28 | 0 | 1 | 8 | 33 | 263 |
| MntGoldProds | 44.02 | 52.17 | 0 | 9 | 24 | 56 | 362 |
| NumDealsPurchases | 2.33 | 1.93 | 0 | 1 | 2 | 3 | 15 |
| NumWebPurchases | 4.08 | 2.78 | 0 | 2 | 4 | 6 | 27 |
| NumCatalogPurchases | 2.66 | 2.92 | 0 | 0 | 2 | 4 | 28 |
| NumStorePurchases | 5.79 | 3.25 | 0 | 3 | 5 | 8 | 13 |
| NumWebVisitsMonth | 5.32 | 2.43 | 0 | 3 | 6 | 7 | 20 |
| AcceptedCmp3 | 0.07 | 0.26 | 0 | 0 | 0 | 0 | 1 |
| AcceptedCmp4 | 0.07 | 0.26 | 0 | 0 | 0 | 0 | 1 |
| AcceptedCmp5 | 0.07 | 0.26 | 0 | 0 | 0 | 0 | 1 |
| AcceptedCmp1 | 0.06 | 0.25 | 0 | 0 | 0 | 0 | 1 |
| AcceptedCmp2 | 0.01 | 0.11 | 0 | 0 | 0 | 0 | 1 |
| Response | 0.15 | 0.36 | 0 | 0 | 0 | 0 | 1 |
| Complain | 0.01 | 0.10 | 0 | 0 | 0 | 0 | 1 |
| Years\_as\_Cust | 6.97 | 0.68 | 6 | 7 | 7 | 7 | 8 |
| Ages | 51.19 | 11.98 | 24 | 43 | 50 | 61 | 127 |
| MntTotal | 301.86 | 338.26 | 1 | 42 | 142.5 | 486.25 | 1729 |
| TotalPurchases | 14.86 | 7.68 | 0 | 8 | 15 | 21 | 44 |
| ChildHome | 0.72 | 0.45 | 0 | 0 | 1 | 1 | 1 |
| UsedDeal | 0.98 | 0.14 | 0 | 1 | 1 | 1 | 1 |

Table : The description of all variables in the dataset

In this dataset, there are some missing values for the income variable and there are some errors in the age of customers (Ex. 127 years old customer is quite not realistic).

As our customers’ profiles are varied and segmented in long ranges of age, income,… so there are many outliers detected during the data exploration steps. I decided to impute the missing values by median imputation instead of mean imputation to avoid biases. This method can be used for other variables in the dataset as well. For values that are out of the tolerance intervals, we can use the cap and floor method:

* We can “floor” all the values that are lower than the lower whisker = quartile 1 - (1.5 \* interquartile)
* And “cap” all the values that are greater than the upper\_whisker = quartile 3 + (1.5 \* interquartile).

As this is the sales data and we are growing stable so this data source should be stable and could only change in the range of +- 8% (based on the historical data to calculate the Compound Annual Growth Rate) and this model is used to predict the result of the marketing campaigns during normal context only. In case of incidence like the Covid-19 pandemic happen, we should expect the sale drop and the number of purchases drop across the channel which higher 8%, we should not use this model. But the profile of customers should not be drifted much.

As we have compared the accuracy and F1 Scores of all the applied models and decided to choose the go with the Logistic Regression with less complex input features. I used another method to measure the performance of the classifier models called ROC curve or Receiver Operating Characteristic which shows the relation between TPR or the proportion positive correctly classified (True Positive/ Positive) and False positive rate (FPR) is calculated by 1-TNR (True Negative/Negative). The ROC region plot TPR against FPR:

from sklearn import datasets

from sklearn.metrics import roc\_curve, roc\_auc\_score

from sklearn.model\_selection import train\_test\_split

import matplotlib.pyplot as plt

false\_positive\_rate\_lored, true\_positive\_rate\_lored, threshold\_lored = roc\_curve(y\_test, y\_pred\_lored)

false\_positive\_rate\_maxtree, true\_positive\_rate\_maxtree, threshold\_maxtree = roc\_curve(y\_test, y\_pred\_maxtree)

false\_positive\_rate\_optree, true\_positive\_rate\_optree, threshold\_optree = roc\_curve(y\_test, y\_pred\_optree)

false\_positive\_rate\_GBC, true\_positive\_rate\_GBC, threshold\_GBC = roc\_curve(y\_test, y\_pred\_GBC)

false\_positive\_rate\_hgb, true\_positive\_rate\_hgb, threshold\_hgb = roc\_curve(y\_test, y\_pred\_hgb)

false\_positive\_rate\_24logit, true\_positive\_rate\_24logit, threshold\_24logit = roc\_curve(y\_test, y\_pred\_24logit)

print('roc\_auc\_score for Logistic Regression: ', roc\_auc\_score(y\_test, y\_pred\_lored))

print('roc\_auc\_score for Maximal Tree: ', roc\_auc\_score(y\_test, y\_pred\_maxtree))

print('roc\_auc\_score for Optimal Tree: ', roc\_auc\_score(y\_test, y\_pred\_optree))

print('roc\_auc\_score for Gradient Boosting Classifer: ', roc\_auc\_score(y\_test, y\_pred\_GBC))

print('roc\_auc\_score for Hyperparameter GBC: ', roc\_auc\_score(y\_test, y\_pred\_hgb))

print('roc\_auc\_score for Logistic Regression with 24 features: ', roc\_auc\_score(y\_test, y\_pred\_24logit))

#Plot the curve

plt.subplots(1, figsize=(10,10))

plt.title('Receiver Operating Characteristic')

plt.plot(false\_positive\_rate\_lored, true\_positive\_rate\_lored)

plt.plot(false\_positive\_rate\_maxtree, true\_positive\_rate\_maxtree)

plt.plot(false\_positive\_rate\_optree, true\_positive\_rate\_optree)

plt.plot(false\_positive\_rate\_GBC, true\_positive\_rate\_GBC)

plt.plot(false\_positive\_rate\_hgb, true\_positive\_rate\_hgb)

plt.plot(false\_positive\_rate\_24logit, true\_positive\_rate\_24logit)

plt.legend(['Logistic Regression', 'Maximal Tree', 'Optimal Tree', 'Gradient Boosting Classifer', 'Hyperparameter GBC', 'Logistic Regression with 24 features' ])

plt.plot([0, 1], ls="--")

plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")

plt.ylabel('True Positive Rate')

plt.xlabel('False Positive Rate')

plt.show()

Output:

roc\_auc\_score for Logistic Regression: 0.7019873353929752

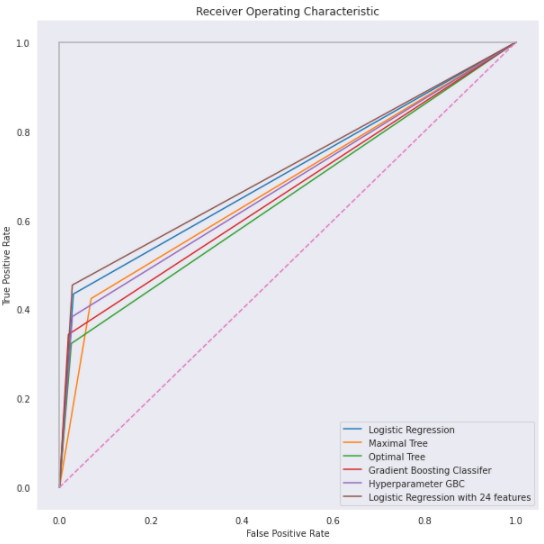
roc\_auc\_score for Maximal Tree: 0.6774140537698021

roc\_auc\_score for Optimal Tree: 0.6486009772343829

roc\_auc\_score for Gradient Boosting Classifer: 0.6619557834308376

roc\_auc\_score for Hyperparameter GBC: 0.6778194088389317

roc\_auc\_score for Logistic Regression with 24 features: 0.713172944192467



Code 21: ROC curves for all models

The more the curve bend to the top left, the better the model is. So it proved again that the Logistic Model with 24 features is performing the best versus the other models. We can use these Roc Scores and Roc Curves to compare the result for future data in the same way of using the RMSE.

[Figure 1: Creating 3 empty tables to import data to MySQL 6](#_Toc90628257)

[Figure 2: Importing data into 3 empty tables 8](#_Toc90628258)

[Figure 3: Merging 3 tables into 1 table 9](#_Toc90628259)

[Figure 4: Exporting final data table for further analysis 9](#_Toc90628260)

[Figure 5: Profiles and behaviors of H&H group vs average 48](#_Toc90628261)

[Figure 6: Profiles and behaviors of L&L group vs average 49](#_Toc90628262)

[Figure 7: Profiles and behaviors of M&M group vs average 50](#_Toc90628263)

[Table 1: List of data variables and details. 5](#_Toc90628264)

[Table 2: Logistic Regression Result summary for 19 Features fitting 46](#_Toc90628265)

[Table 3: The description of all variables in the dataset 55](#_Toc90628266)

[Code 1: Data importing and overview 11](#_Toc90628267)

[Code 2: Exploring “Income” variable 11](#_Toc90628268)

[Code 3: Applying median imputation to missing values in the “Income” column 12](#_Toc90628269)

[Code 4: Engineering “Years\_as\_Cust” column 13](#_Toc90628270)

[Code 5: Engineering “Age” column 13](#_Toc90628271)

[Code 6: Engineering “MntTotal” and “TotalPurchases” column 13](#_Toc90628272)

[Code 7: Engineering indicator for the household with child and customer with deal purchase column 14](#_Toc90628273)

[Code 8: Apply cap & floor transformation for all outliers 18](#_Toc90628274)

[Code 9: Data scaling using Standard Scaler 20](#_Toc90628275)

[Code 10: Transforming categorical variables into dummies variables 22](#_Toc90628276)

[Code 11: Correlation between variables 23](#_Toc90628277)

[Code 12: Data partition 27](#_Toc90628278)

[Code 13: Apply Logistic Regression 29](#_Toc90628279)

[Code 14: Apply Maximal Decision Tree 31](#_Toc90628280)

[Code 15: Apply optimal Decision Tree 34](#_Toc90628281)

[Code 16: Apply Gradient Boosting Classifier 38](#_Toc90628282)

[Code 17: Apply Neural Network 39](#_Toc90628283)

[Code 18: Applying RFE to find the significant variables 44](#_Toc90628284)

[Code 19: Response numbers and rates by cluster 45](#_Toc90628285)

[Code 20: Summary of Logistic Regression with 24 features 53](#_Toc90628286)

[Code 21: ROC curves for all models 59](#_Toc90628287)