# Tarea 3 Final

June 14, 2023

## 1 Tarea_3_LAB_MAA_Diego_Valdebenito

**Tarea 3**

*Instrucciones*

Los resultados de los ejercicios propuestos se deben entregar como un notebook por correo electronico a *juancaros@udec.cl* el dia 9/6 hasta las 21:00. Es importante considerar que el código debe poder ejecutarse en cualquier computadora con la data original del repositorio. Recordar la convencion para el nombre de archivo ademas de incluir en su documento titulos y encabezados por seccion. Utilizar la base de datos *ifood_df.csv*.

Como se indica en la Tabla 1, las variables describen el comportamiento de un set de consumidores en una tienda de retail. Las variables categoricas (e.g. educacion, estado civil) ya han sido convertidas a variables binarias (una por cada categoria).

Preguntas:

1. Cargue la base de datos y realice los ajustes necesarios para su uso (missing values, recodificar variables, etcetera). Identifique los tipos de datos que se encuentran en la base, realice estadisticas descriptivas sobre las variables importantes (Hint: Revisar la distribuciones, datos faltantes, outliers, etc.) y limpie las variables cuando sea necesario.
2. Realice un PCA usando las variables de numero de compras y cantidad gastada en los diversos items. En particular, identifique los valores propios y determine el numero optimo de componentes. Luego estime y grafique la distribucion de los componentes. Ademas discuta la importancia relativa de las variables sobre cada uno de los componentes estimados. Que se puede concluir de este analisis?
3. Con los resultados de la Pregunta 2, mantenga los primeros 3 componentes principales y repita el analisis. Graficamente y estadisticamente indique si existen diferencias o relaciones significativas entre los valores de los PCA y las siguientes variables: Income, Kidhome, Education y Recency. Que puede concluir de los resultados?
4. A partir del mismo set de variables de la pregunta 2 realice un EFA. En particular determine el numero optimo de factores y las variables que se asocian a cada factor. Tambien discuta si existen variables que no son informativas.
5. Con los resultados obtenidos en la Pregunta 4, proponga un CFA donde cada variable solo se asocia con un factor. Entregue un nombre a cada factor que representa el concepto comun entre todas las variables. Reporte la importancia de cada medida (variable) a cada factor e indique la correlacion entre factores.

6. Finalmente, implemente un SEM completo usando la estructura propuesta en la Pregunta 5. En particular, estime un modelo donde los factores explican la variable Response, junto con otras variables demograficas que existen en la base de datos. Ademas utilice dichas variables relevantespara explicar los factores latentes si lo considera apropiado. Las variables a incluir en el modelo final deben tener sustento teorico y el modelo final debe optimizar el ajuste a los datos, en base a los criterios vistos en clase. Que puede concluir en base a sus resultados?

### 1.1 Librerias a utilizar

[5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.formula.api as smf
import sklearn
import scipy
from scipy.linalg import eigh, cholesky
from scipy.stats import norm
import linearmodels.panel as lmp
from pylab import plot, show, axis, subplot, xlabel, ylabel, grid
import semopy
import seaborn as sns
from factor_analyzer import FactorAnalyzer
from sklearn.decomposition import PCA
from IPython.display import Image
from PIL import Image



%matplotlib inline
```

[6]:
```
imagen = Image.open("C:/Users/equipo/Desktop/Tarea_3_Laboratorio/dictionary.
↪png")
imagen.show()
```

First, we simulate some data to use all different methods. Nine continuous variables and nine binary variables, all with different levels of correlation among them.

[7]:
```
df_food=pd.read_csv("C:/Users/equipo/Desktop/Tarea_3_Laboratorio/ifood_df.csv")
#data description
df=df_food[['Income','Kidhome','Teenhome','Recency','MntWines','MntFruits','MntMeatProducts','
           ␣
↪'MntSweetProducts','MntGoldProds','NumDealsPurchases','NumWebPurchases',
           ␣
↪'NumCatalogPurchases','NumStorePurchases','NumWebVisitsMonth','AcceptedCmp3','AcceptedCmp4'
           ␣
↪'AcceptedCmp2','Complain','Z_CostContact','Z_Revenue','Response','Age','Customer_Days','mar
```

```
            'marital_Single','marital_Together','marital_Widow','education_2n␣
 ↪Cycle','education_Basic','education_Graduation','education_Master','education_PhD','MntTota
            'MntRegularProds','AcceptedCmpOverall']]
df.describe()
```

[7]:

| | Income | Kidhome | Teenhome | Recency | MntWines |
|---|---|---|---|---|---|
| count | 2205.000000 | 2205.000000 | 2205.000000 | 2205.000000 | 2205.000000 |
| mean | 51622.094785 | 0.442177 | 0.506576 | 49.009070 | 306.164626 |
| std | 20713.063826 | 0.537132 | 0.544380 | 28.932111 | 337.493839 |
| min | 1730.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 35196.000000 | 0.000000 | 0.000000 | 24.000000 | 24.000000 |
| 50% | 51287.000000 | 0.000000 | 0.000000 | 49.000000 | 178.000000 |
| 75% | 68281.000000 | 1.000000 | 1.000000 | 74.000000 | 507.000000 |
| max | 113734.000000 | 2.000000 | 2.000000 | 99.000000 | 1493.000000 |

| | MntFruits | MntMeatProducts | MntFishProducts | MntSweetProducts |
|---|---|---|---|---|
| count | 2205.000000 | 2205.000000 | 2205.000000 | 2205.000000 |
| mean | 26.403175 | 165.312018 | 37.756463 | 27.128345 |
| std | 39.784484 | 217.784507 | 54.824635 | 41.130468 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 2.000000 | 16.000000 | 3.000000 | 1.000000 |
| 50% | 8.000000 | 68.000000 | 12.000000 | 8.000000 |
| 75% | 33.000000 | 232.000000 | 50.000000 | 34.000000 |
| max | 199.000000 | 1725.000000 | 259.000000 | 262.000000 |

| | MntGoldProds | … | marital_Together | marital_Widow | education_2n Cycle |
|---|---|---|---|---|---|
| count | 2205.000000 | … | 2205.000000 | 2205.000000 | 2205.000000 |
| mean | 44.057143 | … | 0.257596 | 0.034467 | 0.089796 |
| std | 51.736211 | … | 0.437410 | 0.182467 | 0.285954 |
| min | 0.000000 | … | 0.000000 | 0.000000 | 0.000000 |
| 25% | 9.000000 | … | 0.000000 | 0.000000 | 0.000000 |
| 50% | 25.000000 | … | 0.000000 | 0.000000 | 0.000000 |
| 75% | 56.000000 | … | 1.000000 | 0.000000 | 0.000000 |
| max | 321.000000 | … | 1.000000 | 1.000000 | 1.000000 |

| | education_Basic | education_Graduation | education_Master | education_PhD |
|---|---|---|---|---|
| count | 2205.000000 | 2205.000000 | 2205.000000 | 2205.000000 |
| mean | 0.024490 | 0.504762 | 0.165079 | 0.215873 |
| std | 0.154599 | 0.500091 | 0.371336 | 0.411520 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 50% | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| 75% | 0.000000 | 1.000000 | 0.000000 | 0.000000 |
| max | 1.000000 | 1.000000 | 1.000000 | 1.000000 |

| | MntTotal | MntRegularProds | AcceptedCmpOverall |
|---|---|---|---|
| count | 2205.000000 | 2205.000000 | 2205.00000 |

```
mean     562.764626       518.707483              0.29932
std      575.936911       553.847248              0.68044
min        4.000000      -283.000000              0.00000
25%       56.000000        42.000000              0.00000
50%      343.000000       288.000000              0.00000
75%      964.000000       884.000000              0.00000
max     2491.000000      2458.000000              4.00000

[8 rows x 39 columns]
```

### 1.1.1 Pregunta 1

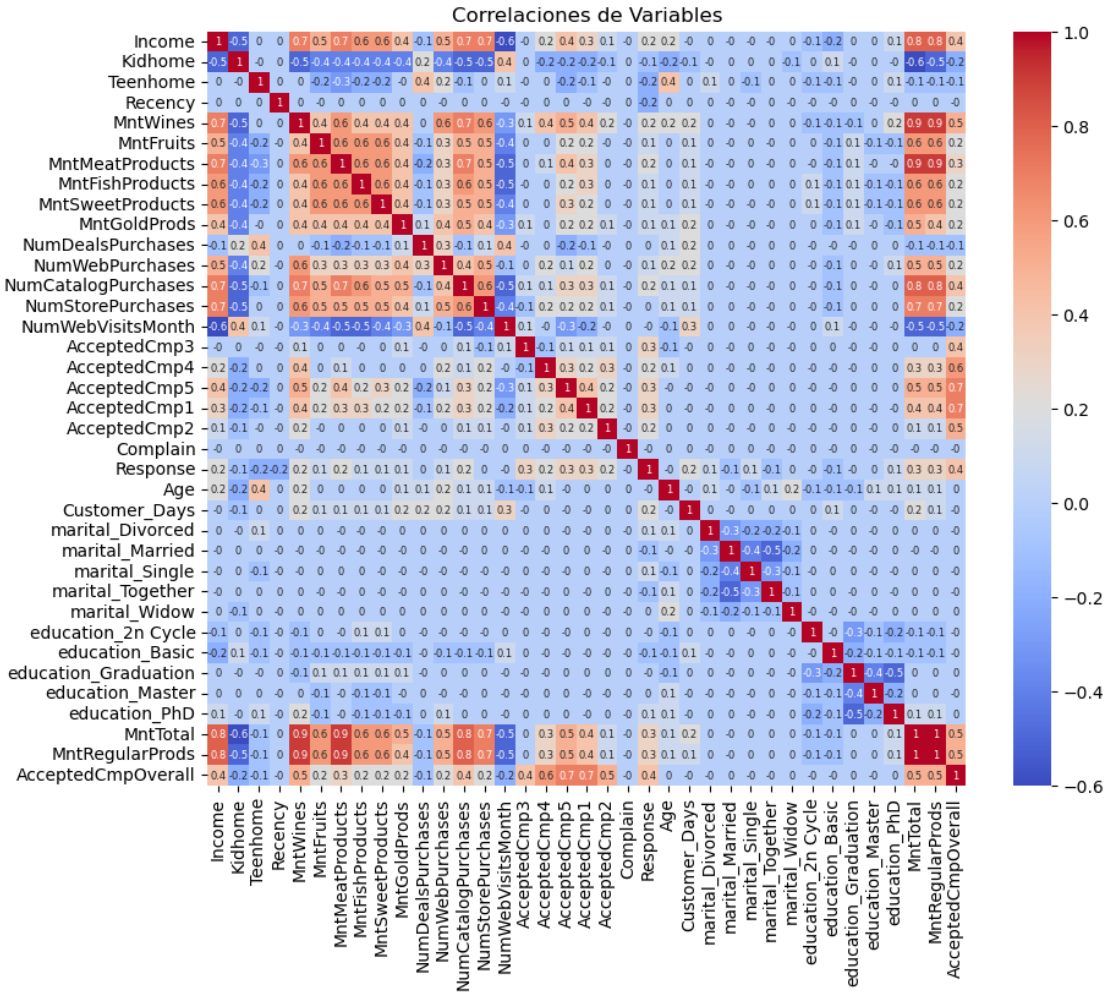
Cabe señalar desde un principio que, para todas las preguntas, el nivel de significatividad utilizado es el clásico, es decir, el de 0.05 o 5%.

### 1.1.2 Multicolinealidad

En la siguiente figura, se muestran las correlaciones entre las variables, se pondrán en el modelo todas las variables que no tengan problemas de multicolinealidad.

[8]:
```
# Importar librerías necesarias
import matplotlib.pyplot as plt
import seaborn as sns

# Seleccionar las variables de interés
variables = ['Income', 'Kidhome', 'Teenhome', 'Recency', 'MntWines',␣
 ↪'MntFruits', 'MntMeatProducts', 'MntFishProducts',
            'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',␣
 ↪'NumWebPurchases', 'NumCatalogPurchases',
            'NumStorePurchases', 'NumWebVisitsMonth', 'AcceptedCmp3',␣
 ↪'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
            'AcceptedCmp2', 'Complain', 'Response', 'Age', 'Customer_Days',␣
 ↪'marital_Divorced', 'marital_Married',
            'marital_Single', 'marital_Together', 'marital_Widow',␣
 ↪'education_2n Cycle', 'education_Basic',
            'education_Graduation', 'education_Master', 'education_PhD',␣
 ↪'MntTotal', 'MntRegularProds',
            'AcceptedCmpOverall']

df = df_food[variables]

# Calcular las correlaciones
correlations = df.corr()

# Crear un heatmap de las correlaciones
plt.figure(figsize=(10, 8))
```

```
sns.heatmap(correlations.round(1), annot=True, cmap='coolwarm',␣
 ↪annot_kws={"fontsize": 6})
plt.title('Correlaciones de Variables')
plt.show()
```




La variable “MntTotal” se decide no ocuparla en los modelos, correspondiente al monto total gastado en los distintos productos, ya que está muy correlacionada con la mayoría de las variables correspondientes al número de compras. Debido a que ocurre algo muy similar con la variable “MntRegularProds”, también no se considerara a esta variable.

[9]:
```
print(df_food.dtypes)

# Detectar variables binarias en el DataFrame "charls"
binaries = []
for column in df_food.columns:
    if df_food[column].nunique() == 2:
```

```
        binaries.append(column)

# Imprimir las variables binarias
print("Variables binarias en la base:")
print(binaries)
```

```
Income                  float64
Kidhome                   int64
Teenhome                  int64
Recency                   int64
MntWines                  int64
MntFruits                 int64
MntMeatProducts           int64
MntFishProducts           int64
MntSweetProducts          int64
MntGoldProds              int64
NumDealsPurchases         int64
NumWebPurchases           int64
NumCatalogPurchases       int64
NumStorePurchases         int64
NumWebVisitsMonth         int64
AcceptedCmp3              int64
AcceptedCmp4              int64
AcceptedCmp5              int64
AcceptedCmp1              int64
AcceptedCmp2              int64
Complain                  int64
Z_CostContact             int64
Z_Revenue                 int64
Response                  int64
Age                       int64
Customer_Days             int64
marital_Divorced          int64
marital_Married           int64
marital_Single            int64
marital_Together          int64
marital_Widow             int64
education_2n Cycle        int64
education_Basic           int64
education_Graduation      int64
education_Master          int64
education_PhD             int64
MntTotal                  int64
MntRegularProds           int64
AcceptedCmpOverall        int64
dtype: object
Variables binarias en la base:
```

```
['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1', 'AcceptedCmp2',
'Complain', 'Response', 'marital_Divorced', 'marital_Married', 'marital_Single',
'marital_Together', 'marital_Widow', 'education_2n Cycle', 'education_Basic',
'education_Graduation', 'education_Master', 'education_PhD']
```

### 1.1.3 Tipos de variables y variables a analizar

Tal como se ordena en el código anterior, se imprimen los tipos de variables de la base de datos, donde todas son enteras, de las cuales "AcceptedCmp3", "AcceptedCmp4", "AcceptedCmp5", "AcceptedCmp1", "AcceptedCmp2", "Complain", "Response", "marital_Divorced", "marital_Married", "marital_Single", "marital_Together", "marital_Widow", "education_2n Cycle", "education_Basic", "education_Graduation", "education_Master" y "education_PhD", son binarias. Debido a que hubo indicios de multicolinealidad anteriormente, no se consideraran las variables "Income", "MntTotal" y "MntRegularProds".

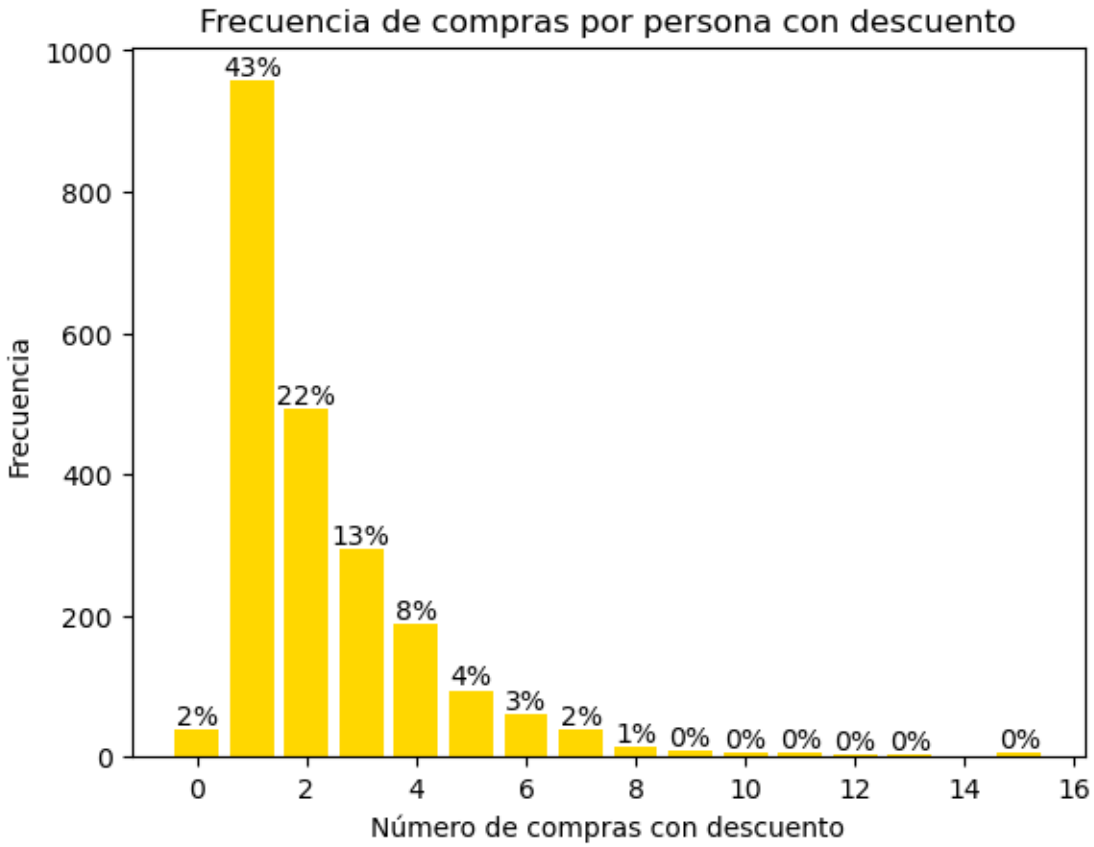
Las variables que se consideran a priori son, "Recency", para saber cuál es la cantidad de días que las personas pasan sin comprar, "NumDealsPurchases", "NumWebPurchases", "NumCatalogPurchases", "NumStorePurchases", "NumWebVisitsMonth"; debido a que se desea saber el número de compras que se realizan según las distintas modalidades de compra y la cantidad de veces en las que es visitado el sitio web al mes, "Response", debido a que es de interés si el target acepto la última campaña publicitaria, "Age" (edad), debido a que se desea saber cual es la predominancia de edad entre los individuos, "Customer_Days" y "Recency", ya que es de interes de cualquier empresa saber los dias desde la primera compra de cada cliente y conocer cuantos dias pasan las personas sin comprar.

A continuación, se muestran los estadísticos descriptivos y gráficamente los comportamientos variables de estudio y explicativas antes mencionadas:

### 1.1.4 NumDealsPurchases

[10]:
```
# Contar la frecuencia
num_deals_count = df_food["NumDealsPurchases"].value_counts()

# Crear un gráfico de barras con los valores en el eje x y la frecuencia en el␣
 ↪eje y
plt.bar(x=num_deals_count.index, height=num_deals_count.values, color="gold")

# Añadir etiquetas al gráfico
plt.title("Frecuencia de compras por persona con descuento")
plt.xlabel("Número de compras con descuento")
plt.ylabel("Frecuencia")

# Agregar etiquetas de porcentaje a cada barra
total_count = len(df_food)
for i, value in enumerate(num_deals_count.values):
    percent_label = "{:.0f}%".format(value / total_count * 100)
    plt.text(num_deals_count.index[i], value, percent_label, ha='center',␣
 ↪va='bottom')
```

```
# Mostrar el gráfico
plt.show()
```

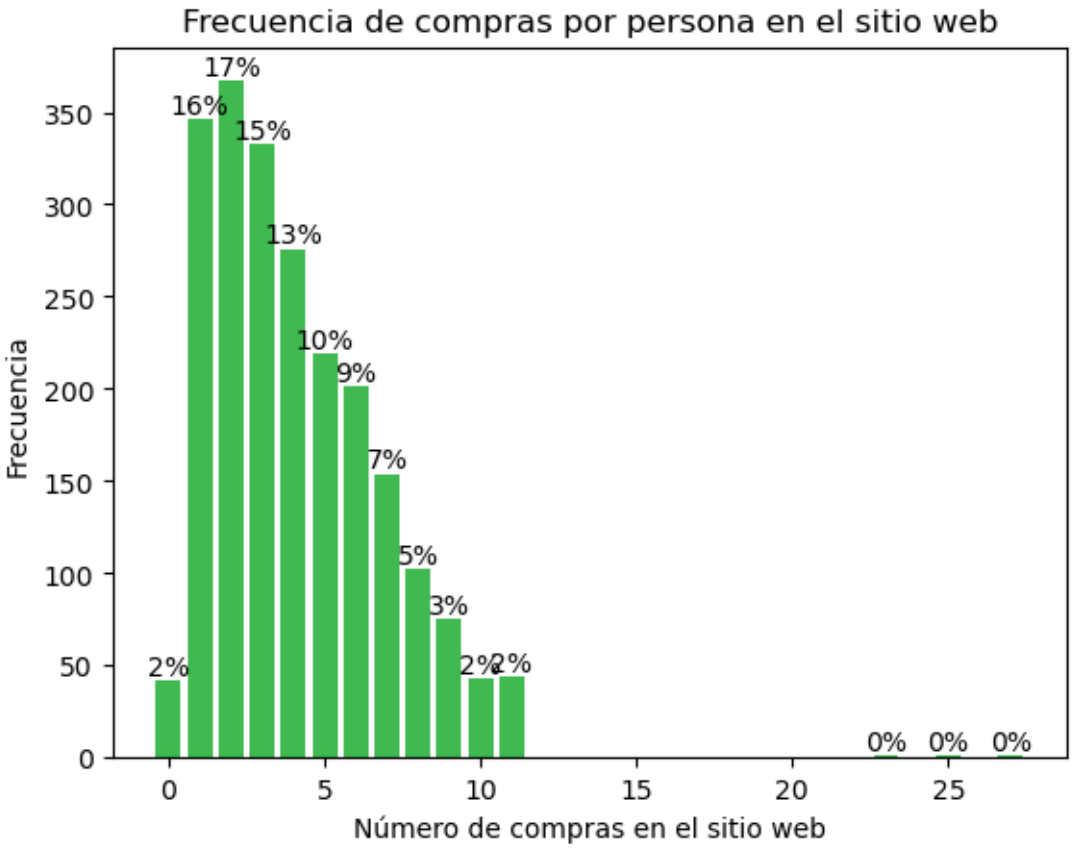



Se puede visualizar claramente que la mayoría de las personas realizaron solo una compra con descuento, siguiéndoles los que realizaron 2, y en menor porcentaje, 3; por lo cual podría ser necesario que las ofertas sean más potentes para estimular un mayor consumo de productos por parte de cada consumidor cuando los productos están con descuento.

### 1.1.5 NumWebPurchases

[11]:
```
# Contar la frecuencia
num_web_count = df_food["NumWebPurchases"].value_counts()

# Crear un gráfico de barras con los valores en el eje x y la frecuencia en el␣
 ↪eje y
plt.bar(x=num_web_count.index, height=num_web_count.values, color="#3FB950")

# Añadir etiquetas al gráfico
plt.title("Frecuencia de compras por persona en el sitio web")
plt.xlabel("Número de compras en el sitio web")
```

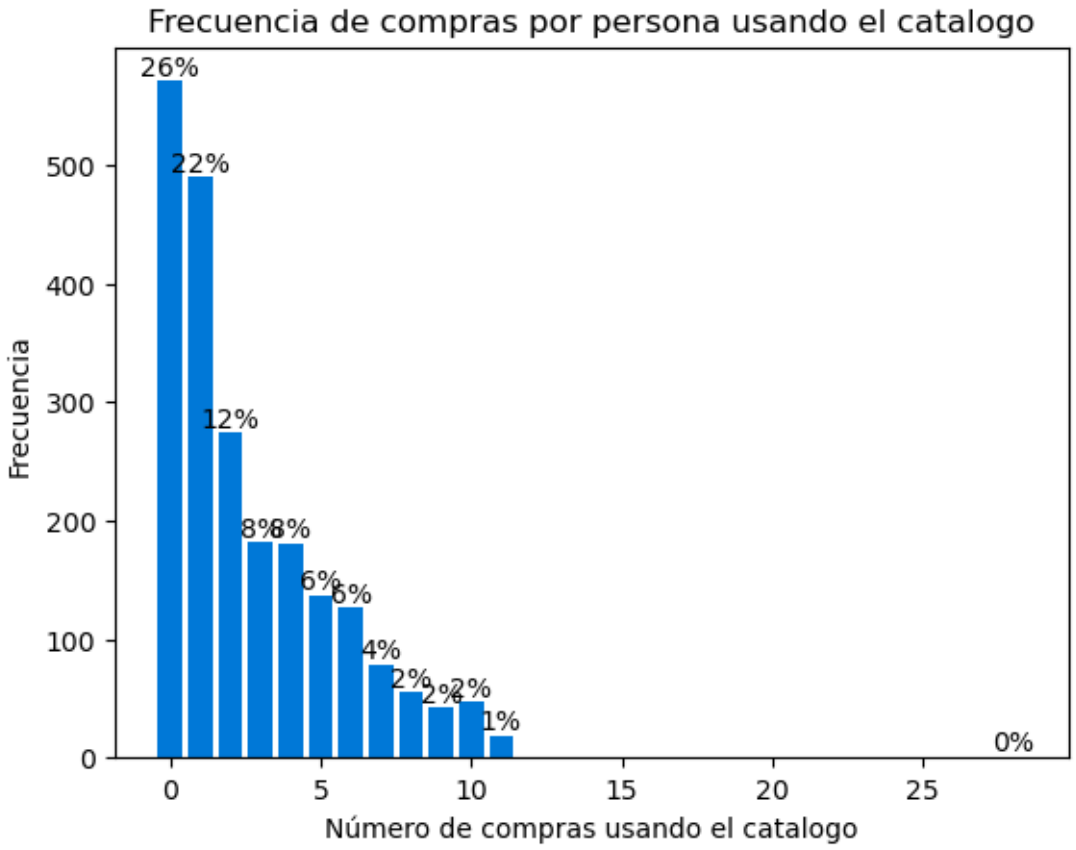
```
plt.ylabel("Frecuencia")

# Agregar etiquetas de porcentaje a cada barra
total_count = len(df_food)
for i, value in enumerate(num_web_count.values):
    percent_label = "{:.0f}%".format(value / total_count * 100)
    plt.text(num_web_count.index[i], value, percent_label, ha='center',␣
 ↪va='bottom')

# Mostrar el gráfico
plt.show()
```




Se puede observar que el número de compras por cliente se distribuye de forma casi equitativa en porcentaje entre 1 y 4 compras, por lo cual es posible afirmar que el sitio web es un éxito para estimular mayor consumo por cada cliente individualmente.

### 1.1.6 NumCatalogPurchases

[12]:
```
# Contar la frecuencia
num_catalog_count = df_food["NumCatalogPurchases"].value_counts()

# Crear un gráfico de barras con los valores en el eje x y la frecuencia en el␣
 ↪eje y
plt.bar(x=num_catalog_count.index, height=num_catalog_count.values,␣
 ↪color="#0078D7")

# Añadir etiquetas al gráfico
plt.title("Frecuencia de compras por persona usando el catalogo")
plt.xlabel("Número de compras usando el catalogo")
plt.ylabel("Frecuencia")

# Agregar etiquetas de porcentaje a cada barra
total_count = len(df_food)
for i, value in enumerate(num_catalog_count.values):
    percent_label = "{:.0f}%".format(value / total_count * 100)
    plt.text(num_catalog_count.index[i], value, percent_label, ha='center',␣
 ↪va='bottom')

# Mostrar el gráfico
plt.show()
```

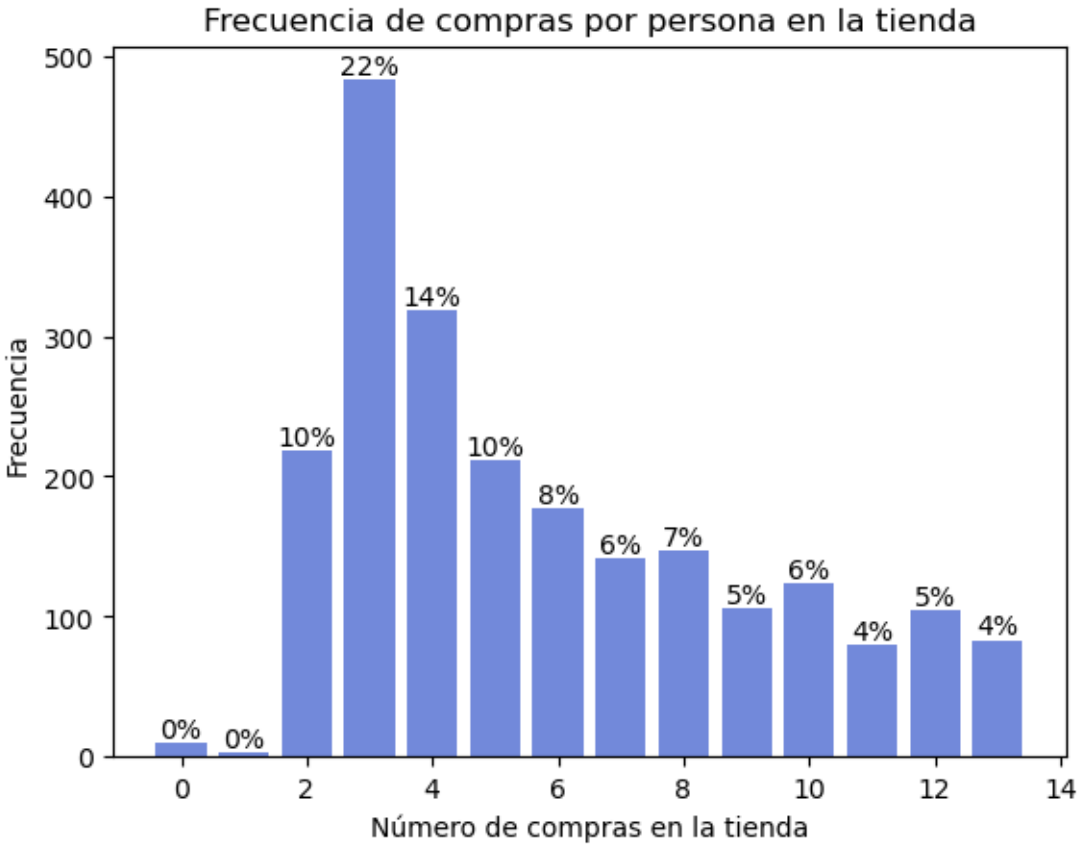



Se puede visualizar que la mayoría de las personas que utilizan el catalogo lo utilizan para cotizar los productos, siguiéndoles los que lo usan solo para comprar un producto y en menor porcentaje, los que compran 2 productos, siendo los que compran más, porcentajes menores. Una buena estrategia podría ser ofrecer algún beneficio por pasar cierto tiempo en el catálogo.

### 1.1.7 NumStorePurchases

[13]:
```
# Contar la frecuencia
num_store_count = df_food["NumStorePurchases"].value_counts()

# Crear un gráfico de barras con los valores en el eje x y la frecuencia en el␣
 ↪eje y
plt.bar(x=num_store_count.index, height=num_store_count.values, color="#7289DA")

# Añadir etiquetas al gráfico
plt.title("Frecuencia de compras por persona en la tienda")
plt.xlabel("Número de compras en la tienda")
plt.ylabel("Frecuencia")

# Agregar etiquetas de porcentaje a cada barra
```

```
total_count = len(df_food)
for i, value in enumerate(num_store_count.values):
    percent_label = "{:.0f}%".format(value / total_count * 100)
    plt.text(num_store_count.index[i], value, percent_label, ha='center',␣
 ↪va='bottom')

# Mostrar el gráfico
plt.show()
```

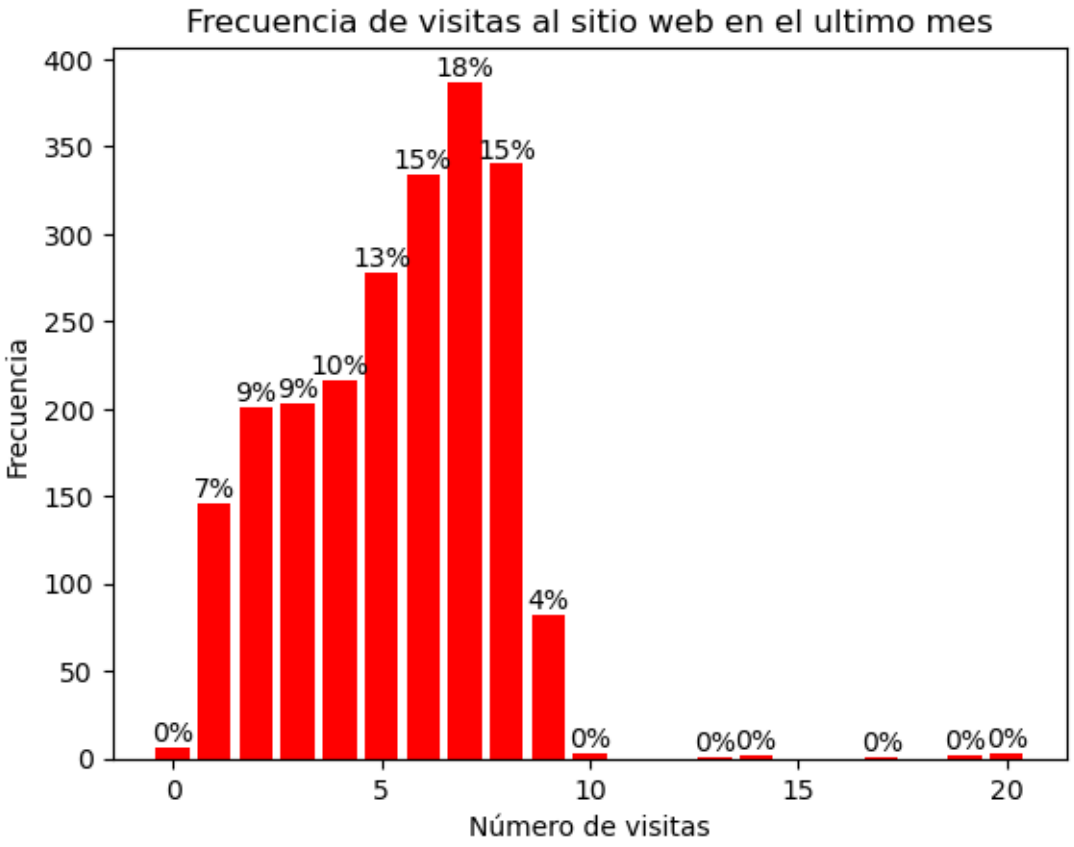



Se puede observar que la mayoría de los individuos que van a la tienda compran 3 productos, siguiéndoles los que compran 4 como predominantes, por lo cual se puede decir que la tienda está siendo un éxito (puede ser debido a un factor como la presentación de esta) para estimular el consumo individual de los clientes.

### 1.1.8 NumWebVisitsMonth

[14]:
```
# Contar la frecuencia
num_vis_count = df_food["NumWebVisitsMonth"].value_counts()
```

```
# Crear un gráfico de barras con los valores en el eje x y la frecuencia en el eje y
plt.bar(x=num_vis_count.index, height=num_vis_count.values, color="#FF0000")

# Añadir etiquetas al gráfico
plt.title("Frecuencia de visitas al sitio web en el ultimo mes")
plt.xlabel("Número de visitas")
plt.ylabel("Frecuencia")

# Agregar etiquetas de porcentaje a cada barra
total_count = len(df_food)
for i, value in enumerate(num_vis_count.values):
    percent_label = "{:.0f}%".format(value / total_count * 100)
    plt.text(num_vis_count.index[i], value, percent_label, ha='center', va='bottom')

# Mostrar el gráfico
plt.show()
```

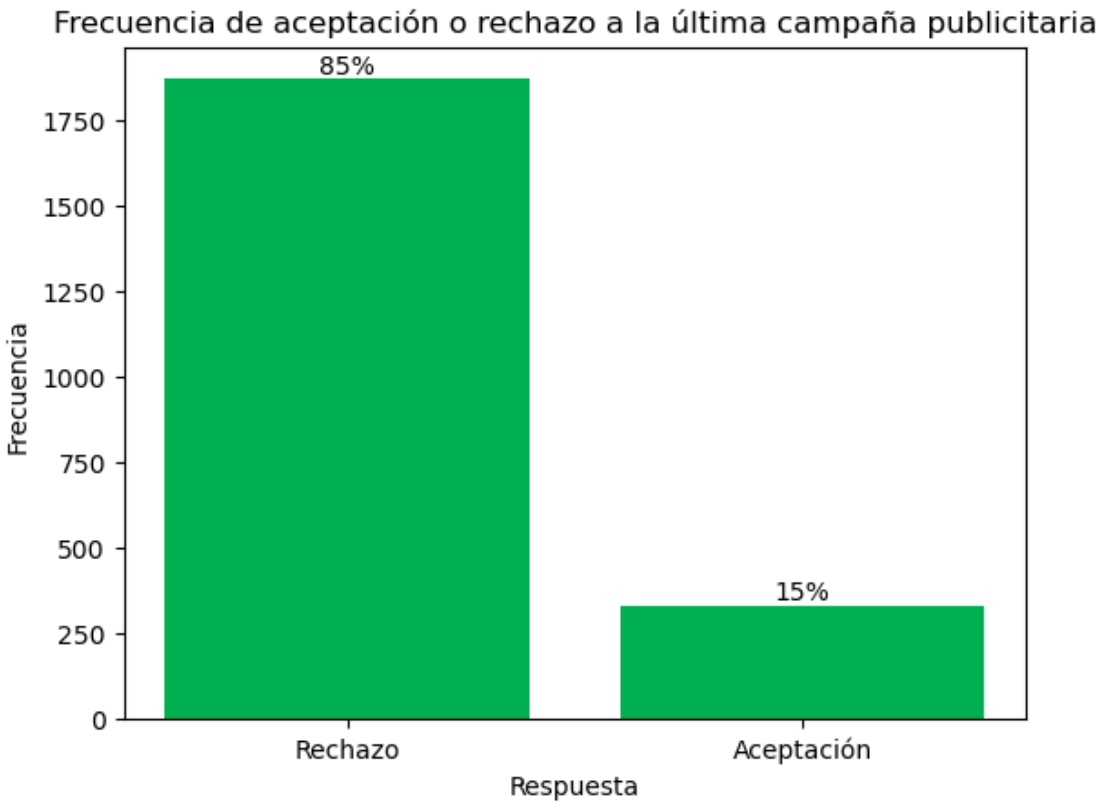



Se puede ver que la mayoría de los individuos visita entre 5 y 9 veces el sitio web, y como se

vio anteriormente, no compran todas estas veces, por lo cual es aconsejable actualizar el sitio web constantemente con nuevos productos o una nueva presentación para que los clientes consideren más atractivo comprar ahí.

### 1.1.9 Response

[15]:
```
# Contar la frecuencia de cada valor en la variable "Response" de la base de␣
 ↪datos "df_food"
num_resp_count = df_food["Response"].value_counts()

# Crear un gráfico de barras con los valores en el eje x y la frecuencia en el␣
 ↪eje y
plt.bar(x=num_resp_count.index, height=num_resp_count.values, color="#00B050")

# Añadir etiquetas al gráfico
plt.title("Frecuencia de aceptación o rechazo a la última campaña publicitaria")
plt.xlabel("Respuesta")
plt.ylabel("Frecuencia")

# Establecer los nuevos labels del eje x
plt.xticks(num_resp_count.index, ['Rechazo', 'Aceptación'])

# Agregar etiquetas de porcentaje a cada barra
total_count = len(df_food)
for i, value in enumerate(num_resp_count.values):
    percent_label = "{:.0f}%".format(value / total_count * 100)
    plt.text(num_resp_count.index[i], value, percent_label, ha='center',␣
 ↪va='bottom')

# Mostrar el gráfico
plt.show()
```



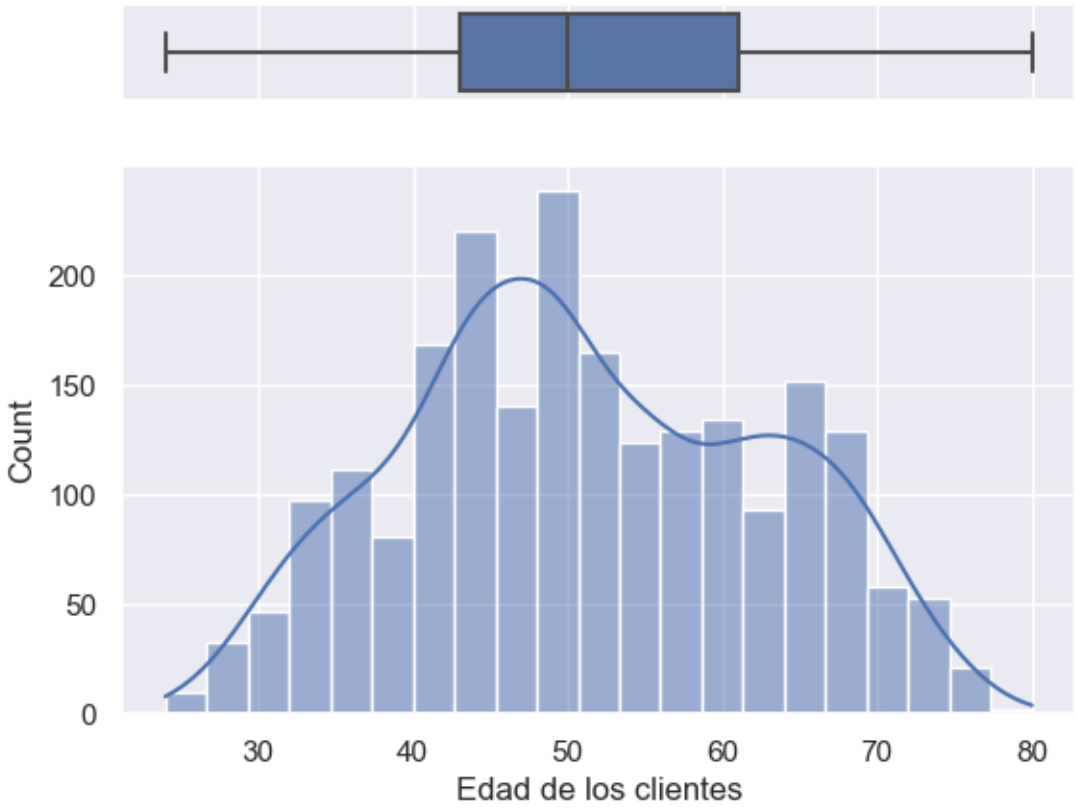

Se puede observar claramente que la gran mayoría del público objetivo mostro su rechazo a la última campaña publicitaria, por lo cual la compañía debería invertir en mejores ideas a la hora de querer promocionar sus productos.

### 1.1.10 Age

[16]:
```
sns.set(style="darkgrid")

# crear una figura compuesta de dos objetos matplotlib.Axes (ax_box y ax_hist)
f, (ax_box, ax_hist) = plt.subplots(2, sharex=True,␣
 ↪gridspec_kw={"height_ratios": (.15, .85)})

# asignar un gráfico a cada objeto ax
sns.boxplot(df_food["Age"], ax=ax_box)
sns.histplot(data=df_food, x="Age", ax=ax_hist, kde=True)
ax_hist.set_xlabel('Edad de los clientes')

# Eliminar el nombre del eje x del boxplot
ax_box.set(xlabel='')

# mostrar el gráfico
```

```
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



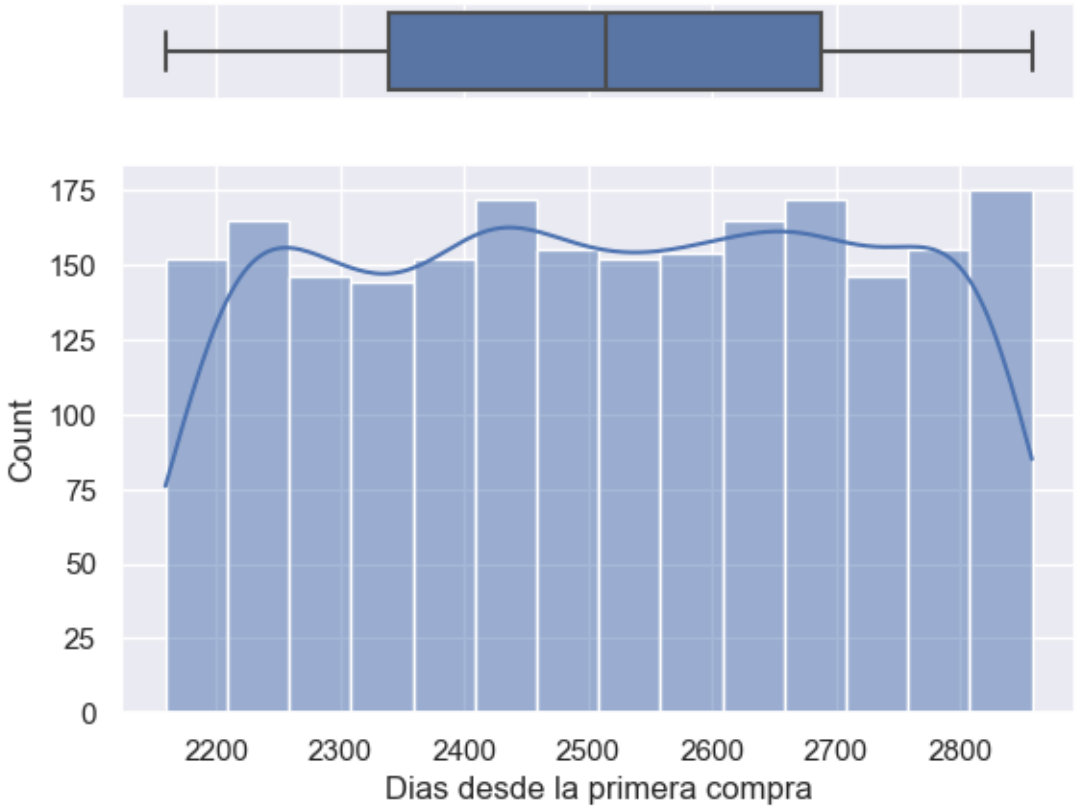

Se puede visualizar que la mayoría de los individuos tiene entre poco más de 40 y 60 años, por lo cual es necesario que tanto los productos como las promociones sean orientados a público de esas edades para que la compañía tenga más éxito en sus ventas.

### 1.1.11 Customer_Days

[17]:
```
sns.set(style="darkgrid")

# crear una figura compuesta de dos objetos matplotlib.Axes (ax_box y ax_hist)
f, (ax_box, ax_hist) = plt.subplots(2, sharex=True,␣
 ↪gridspec_kw={"height_ratios": (.15, .85)})
```

```
# asignar un gráfico a cada objeto ax
sns.boxplot(df_food["Customer_Days"], ax=ax_box)
sns.histplot(data=df_food, x="Customer_Days", ax=ax_hist, kde=True)
ax_hist.set_xlabel('Dias desde la primera compra')

# Eliminar el nombre del eje x del boxplot
ax_box.set(xlabel='')

# mostrar el gráfico
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



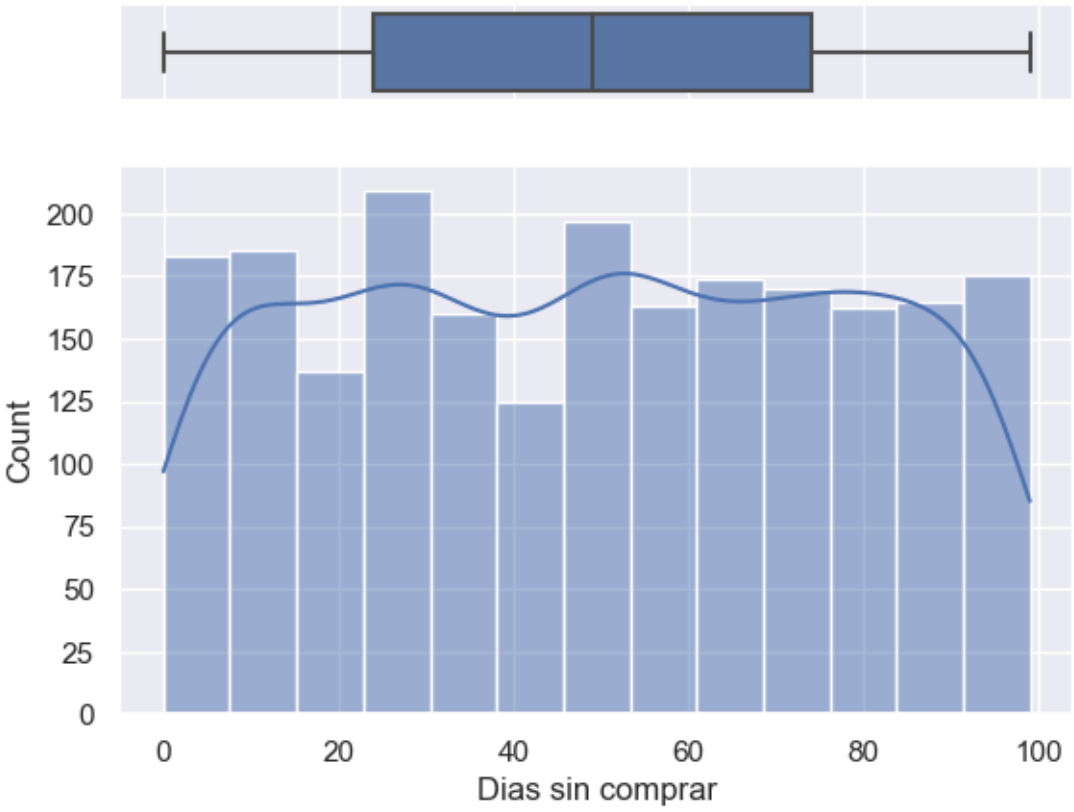

Se observa una distribución prácticamente uniforme con respecto a los días desde la primera compra, entre los 2000 y los 3000 días, sin embargo, la mayoría de las observaciones están entre los 2300 y los 2700 días, por lo cual es aconsejable llegar a los clientes antiguos con más ofertas u ofertas

exclusivas para que sigan comprando los productos de la compañía.

### 1.1.12 Recency

[18]:
```
sns.set(style="darkgrid")

# crear una figura compuesta de dos objetos matplotlib.Axes (ax_box y ax_hist)
f, (ax_box, ax_hist) = plt.subplots(2, sharex=True,␣
 ↪gridspec_kw={"height_ratios": (.15, .85)})

# asignar un gráfico a cada objeto ax
sns.boxplot(df_food["Recency"], ax=ax_box)
sns.histplot(data=df_food, x="Recency", ax=ax_hist, kde=True)
ax_hist.set_xlabel('Dias sin comprar')

# Eliminar el nombre del eje x del boxplot
ax_box.set(xlabel='')

# mostrar el gráfico
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```




Se observa que la mayoría de los clientes pasan entre 20 y 80 días sin comprar, por lo que puede ser necesario que cada semana les llegue una notificación con noticias de los productos para que estas personas deseen volver a consumir.

### 1.1.13 Pregunta 2

A continuación, se procederá a normalizar (estandarizar) las variables de la base de datos para poder aplicar correctamente el "PCA".

[21]:
```
from sklearn.preprocessing import StandardScaler

# Datos de tu DataFrame df_food
X = df_food[['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',
             'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',␣
 ↪'NumWebPurchases', 'NumCatalogPurchases',
             'NumStorePurchases']].values

# Estandarizar los datos
scaler = StandardScaler().fit(X)
rescaledX = scaler.transform(X)
```

```
# Crear un nuevo DataFrame con los datos estandarizados
df_standardized = pd.DataFrame(rescaledX, columns=['MntWines', 'MntFruits',␣
↪'MntMeatProducts', 'MntFishProducts',
            'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',␣
↪'NumWebPurchases', 'NumCatalogPurchases',
            'NumStorePurchases'])

# Imprimir los datos estandarizados
print(df_standardized)
```

```
      MntWines  MntFruits  MntMeatProducts  MntFishProducts  MntSweetProducts  \
0     0.974566   1.548614         1.748400         2.449154          1.480301
1    -0.874776  -0.638664        -0.731678        -0.652345         -0.635399
2     0.355155   0.568110        -0.175957         1.336263         -0.149031
3    -0.874776  -0.563241        -0.667380        -0.506392         -0.586763
4    -0.394659   0.417263        -0.217292         0.150396         -0.003121
…          …          …                …                …                 …
2200  1.193879   0.417263         0.076644         0.077420          2.209853
2201  0.295881  -0.663806        -0.621452        -0.688833         -0.659718
2202  1.783653   0.542969         0.237389        -0.105022         -0.367897
2203  0.361082   0.090428         0.223611         0.770696          0.069834
2204 -0.658427  -0.588382        -0.479078        -0.652345         -0.635399

      MntGoldProds  NumDealsPurchases  NumWebPurchases  NumCatalogPurchases  \
0         0.849556           0.361479         1.424772             2.628526
1        -0.735767          -0.168834        -1.132957            -0.588043
2        -0.039771          -0.699147         1.424772            -0.230646
3        -0.755100          -0.168834        -0.767567            -0.945440
4        -0.561768           1.422105         0.328602             0.126750
…              …                  …                …                    …
2200      3.923536          -0.168834         1.790162             0.126750
2201     -0.697100           2.482731         1.424772            -0.230646
2202     -0.387769          -0.699147        -0.767567             0.126750
2203      0.327560          -0.168834         0.693992             0.841543
2204     -0.445768           0.361479        -0.402177            -0.588043

      NumStorePurchases
0             -0.562650
1             -1.179732
2              1.288596
3             -0.562650
4              0.054432
…                   …
2200          -0.562650
2201          -0.254109
2202           2.214218
2203           1.288596
```

```
2204          -0.562650

[2205 rows x 10 columns]
```

## 1.2 PCA 1

A continuación, se realiza un primer PCA para las variables correspondientes a la cantidad gastada en los distintos productos y al número de compras a través de los diferentes canales considerando 10 componentes en un principio, obteniéndose así, la proporción de la varianza que aporta cada uno de ellas.

### 1.2.1 Proporcion de varianza

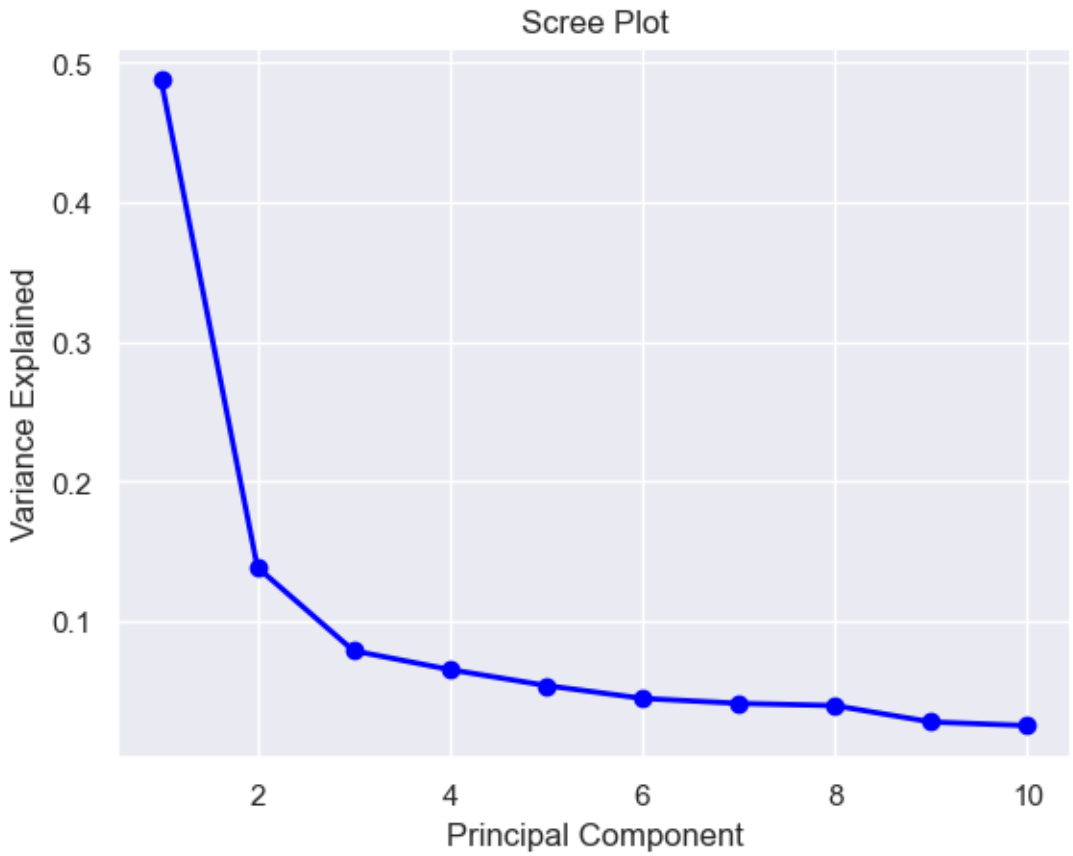
[23]:
```
pca = PCA(n_components=10)
pca_features = pca.fit_transform(df_standardized)
print(pca.explained_variance_ratio_)
```

```
[0.48773352 0.13832059 0.07850407 0.06503529 0.05361044 0.04440343
 0.04082509 0.03921513 0.02747616 0.02487628]
```

Se puede observar claramente que la primera componente explica el 48.77% de la varianza, mientras que la componente 2 explica el 13.83% y la componente 3 explica el 7.85%, mientras que las demás explican menos porcentaje de esta.

### 1.2.2 Scree Plot

[24]:
```
#scree plot using explained variance proportion

PC_values = np.arange(pca.n_components_) + 1
plt.plot(PC_values, pca.explained_variance_ratio_, 'o-', linewidth=2,␣
 ↪color='blue')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()
```




Se puede observar claramente en el "Scree Plot" que solo 2 componentes entregan información, ya que, desde la tercera componente, la varianza explicada es inferior al 10%, luego, la cantidad optima de componentes a priori debería ser de 2.

[31]:
```
pca1 = PCA(n_components=2)
pca1_features = pca1.fit_transform(df_standardized)
print(pca1.explained_variance_ratio_)
```

```
[0.48773352 0.13832059]
```

[32]:
```
print(pca1.explained_variance_ratio_.sum())
```

```
0.626054117197695
```

Podemos ver que en total, las componentes explican el 62.61% de la varianza de las variables.

[33]:
```
pca1_vectors = pd.DataFrame(data = pca1.components_)
pca1_vectors.head()
```

[33]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0.345652 | 0.329467 | 0.367261 | 0.339410 | 0.328118 | 0.279623 | -0.026615 |

```
1   0.209308 -0.228660 -0.180425 -0.237768 -0.204254   0.145564   0.696183

          7         8         9
0  0.273978  0.378615  0.341941
1  0.475368 -0.012596  0.202743
```

### 1.2.3 Valores propios

A continuacion, se calculan los valores propios asociados a cada componente:

[61]:
```
# Calcula la matriz de covarianza de los datos estandarizados
covariance_matrix = np.cov(df_standardized.T)

# Calcula los valores propios
eigenvalues = np.linalg.eigvals(covariance_matrix)

# Imprime los valores propios
print("Valores propios:")
for i, eigenvalue in enumerate(eigenvalues):
    print(f"Componente {i + 1}: {eigenvalue}")
```

```
Valores propios:
Componente 1: 4.87954818664217
Componente 2: 1.383833521256734
Componente 3: 0.7853969141008371
Componente 4: 0.6506479975596254
Componente 5: 0.24887563636531465
Componente 6: 0.2748862449396715
Componente 7: 0.5363476002177077
Componente 8: 0.3923292145368657
Componente 9: 0.40843617036833696
Componente 10: 0.44423571909441545
```

Se considera que los componentes principales 1 y 2 son significativos, ya que tienen valores propios superiores a 1 por lo cual se confirma que estos son los realmente valiosos para el análisis.

[34]:
```
pca_df1 = pd.DataFrame(data=pca_features,columns=['PC1', 'PC2'])
pca_df1.describe().apply(lambda s: s.apply('{0:.3f}'.format))
```

[34]:
```
             PC1       PC2
count  2205.000  2205.000
mean      0.000     0.000
std       2.209     1.176
min      -3.035    -3.001
25%      -2.039    -0.873
50%      -0.578    -0.277
75%       1.829     0.638
max       6.429     5.529
```

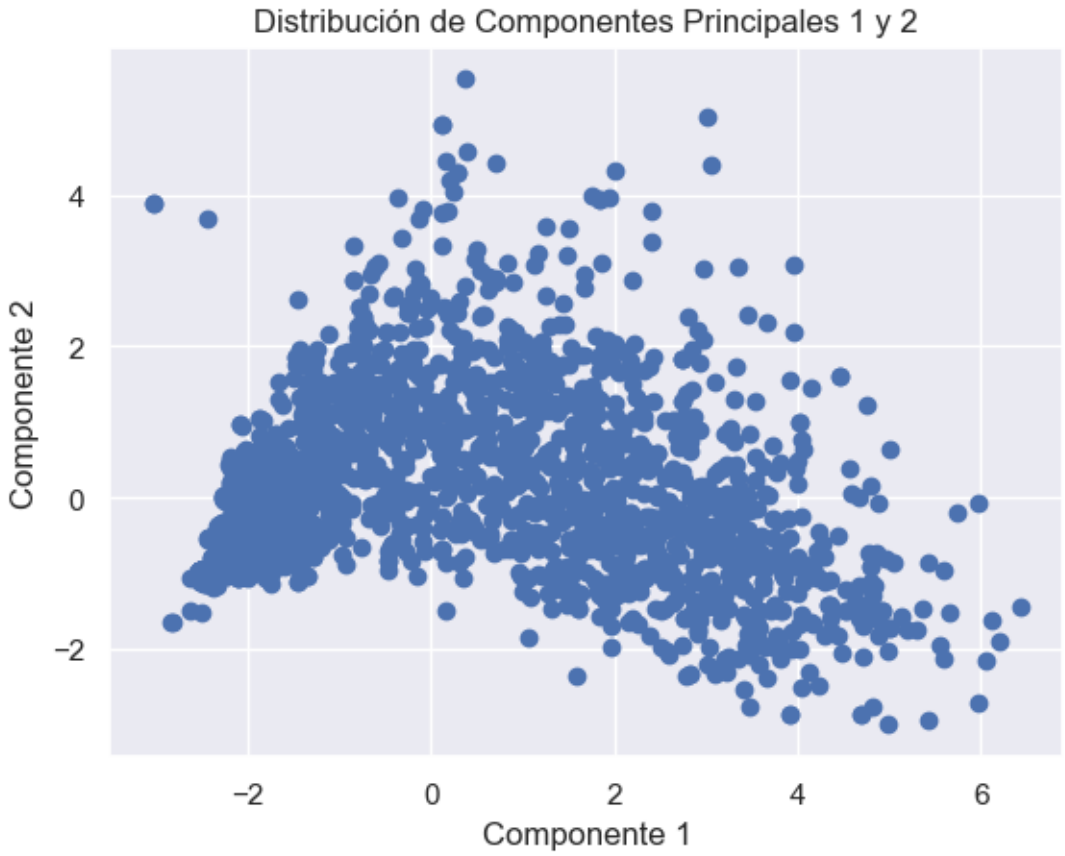
[35]:
```
pca_df1.corr().apply(lambda s: s.apply('{0:.3f}'.format))
```

[35]:

| | PC1 | PC2 |
|---|---|---|
| PC1 | 1.000 | -0.000 |
| PC2 | -0.000 | 1.000 |

Aqui se puede ver a priori que las componentes no estan relacionadas.

[183]:
```
plt.scatter(pca_df1['PC1'],pca_df1['PC2'])
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.title('Distribución de Componentes Principales 1 y 2')
plt.show()
```




Gráficamente, se ve que no hay una tendencia lineal clara entre las 2 componentes, por lo cual es posible afirmar que estas NO están relacionadas entre ellas. Esto indica que las variables no están altamente correlacionadas. A continuación, se muestran las cargas de las variables en cada componente:

[41]:
```
# Variables de interés
variables = ['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',␣
 ↪'MntSweetProducts', 'MntGoldProds',
            'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases',␣
 ↪'NumStorePurchases']

# Obtener las cargas de las variables en cada componente
cargas_variables = pca.components_.T * np.sqrt(pca.explained_variance_)

# Imprimir las cargas de las variables en cada componente
for i, variable in enumerate(variables):
    print(f'Importancia relativa de {variable}:')
    print(f'Componente 1: {cargas_variables[i, 0]}')
    print(f'Componente 2: {cargas_variables[i, 1]}')
```

```
Importancia relativa de MntWines:
Componente 1: 0.7635347324473829
Componente 2: 0.24622279115292708
Importancia relativa de MntFruits:
Componente 1: 0.7277821344337411
Componente 2: -0.2689880171408754
Importancia relativa de MntMeatProducts:
Componente 1: 0.811268847134401
Componente 2: -0.21224614376991344
Importancia relativa de MntFishProducts:
Componente 1: 0.7497461152656371
Componente 2: -0.2797016777676929
Importancia relativa de MntSweetProducts:
Componente 1: 0.7248036344463702
Componente 2: -0.24027703655891933
Importancia relativa de MntGoldProds:
Componente 1: 0.6176792594209969
Componente 2: 0.17123596571231295
Importancia relativa de NumDealsPurchases:
Componente 1: -0.058792282633016286
Componente 2: 0.8189646724155947
Importancia relativa de NumWebPurchases:
Componente 1: 0.6052096961784983
Componente 2: 0.5592065996088456
Importancia relativa de NumCatalogPurchases:
Componente 1: 0.8363498124238129
Componente 2: -0.01481788780126739
Importancia relativa de NumStorePurchases:
Componente 1: 0.7553379313184798
Componente 2: 0.23850013659582345
```

Es posible afirmar tras observar las cargas de las variables en cada componente que en el caso de las variables: “MntWines”, “MntFruits”, “MntMeatProducts”, “MntFishProducts”, “MntSweetProd-

ucts", "MntGoldProds", "NumWebPurchases", "NumCatalogPurchases" y "NumStorePurchases", estas contribuyen más a la componente 1 que a la componente 2; y en el caso de la variable "NumDealsPurchases", esta contribuye más a la componente 2 en comparación a la componente 1. Con esto se quiere decir que entre lo gastado en los distintos productos y el número de compras, solo el número de compras con descuento contribuye más a la componente 2 que a la componente 1.

En resumen, se identificaron las variables que más contribuyen a cada componente y se determinó que el número de compras con descuento tiene una mayor influencia en la segunda componente. En general, las dos primeras componentes explican una parte significativa de la variabilidad en los datos, pero aún hay una cantidad considerable de varianza no explicada por estas dos componentes.

#### 1.2.4 Pregunta 3

### 1.3 PCA 2

A continuación, se realiza el mismo PCA, pero esta vez, considerando 3 componentes.

[150]:
```
from sklearn.preprocessing import StandardScaler

# Datos de tu DataFrame df_food
X2 = df_food[['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',
              'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',␣
 ↪'NumWebPurchases', 'NumCatalogPurchases',
              'NumStorePurchases','Income','Kidhome','education_2n Cycle',
                'education_Basic', 'education_Graduation', 'education_Master',
                'education_PhD','Recency']].values

# Estandarizar los datos
scaler = StandardScaler().fit(X2)
rescaledX2 = scaler.transform(X2)

# Crear un nuevo DataFrame con los datos estandarizados
df_standardized2 = pd.DataFrame(rescaledX2, columns=['MntWines', 'MntFruits',␣
 ↪'MntMeatProducts', 'MntFishProducts',
              'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases',␣
 ↪'NumWebPurchases', 'NumCatalogPurchases',
              'NumStorePurchases','Income','Kidhome','education_2n Cycle',
                'education_Basic', 'education_Graduation', 'education_Master',
                'education_PhD','Recency'])

# Imprimir los datos estandarizados
print(df_standardized2)
```

```
      MntWines  MntFruits  MntMeatProducts  MntFishProducts  MntSweetProducts  \
0     0.974566   1.548614         1.748400         2.449154          1.480301
1    -0.874776  -0.638664        -0.731678        -0.652345         -0.635399
2     0.355155   0.568110        -0.175957         1.336263         -0.149031
3    -0.874776  -0.563241        -0.667380        -0.506392         -0.586763
```

```
4    -0.394659   0.417263        -0.217292         0.150396         -0.003121
…         …          …                …                 …                 …
2200  1.193879   0.417263         0.076644         0.077420          2.209853
2201  0.295881  -0.663806        -0.621452        -0.688833         -0.659718
2202  1.783653   0.542969         0.237389        -0.105022         -0.367897
2203  0.361082   0.090428         0.223611         0.770696          0.069834
2204 -0.658427  -0.588382        -0.479078        -0.652345         -0.635399

      MntGoldProds  NumDealsPurchases  NumWebPurchases  NumCatalogPurchases  \
0         0.849556           0.361479         1.424772             2.628526
1        -0.735767          -0.168834        -1.132957            -0.588043
2        -0.039771          -0.699147         1.424772            -0.230646
3        -0.755100          -0.168834        -0.767567            -0.945440
4        -0.561768           1.422105         0.328602             0.126750
…              …                  …                …                    …
2200      3.923536          -0.168834         1.790162             0.126750
2201     -0.697100           2.482731         1.424772            -0.230646
2202     -0.387769          -0.699147        -0.767567             0.126750
2203      0.327560          -0.168834         0.693992             0.841543
2204     -0.445768           0.361479        -0.402177            -0.588043

      NumStorePurchases    Income   Kidhome  education_2n Cycle  \
0             -0.562650  0.314651 -0.823405           -0.314093
1             -1.179732 -0.254877  1.038757           -0.314093
2              1.288596  0.965354 -0.823405           -0.314093
3             -0.562650 -1.206087  1.038757           -0.314093
4              0.054432  0.322136  1.038757           -0.314093
…                 …        …        …                    …
2200          -0.562650  0.463624 -0.823405           -0.314093
2201          -0.254109  0.598401  2.900920           -0.314093
2202           2.214218  0.258780 -0.823405           -0.314093
2203           1.288596  0.851004 -0.823405           -0.314093
2204          -0.562650  0.060213  1.038757           -0.314093

      education_Basic  education_Graduation  education_Master  education_PhD  \
0           -0.158444              0.990521         -0.444656      -0.524694
1           -0.158444              0.990521         -0.444656      -0.524694
2           -0.158444              0.990521         -0.444656      -0.524694
3           -0.158444              0.990521         -0.444656      -0.524694
4           -0.158444             -1.009570         -0.444656       1.905873
…                 …                     …                 …              …
2200        -0.158444              0.990521         -0.444656      -0.524694
2201        -0.158444             -1.009570         -0.444656       1.905873
2202        -0.158444              0.990521         -0.444656      -0.524694
2203        -0.158444             -1.009570          2.248931      -0.524694
2204        -0.158444             -1.009570         -0.444656       1.905873

       Recency
```

```
0     0.310830
1    -0.380600
2    -0.795458
3    -0.795458
4     1.555404
…        …
2200 -0.104028
2201  0.241687
2202  1.451690
2203 -1.417746
2204 -0.311457

[2205 rows x 18 columns]
```

[151]:
```
pca2 = PCA(n_components=3)
pca2_features = pca2.fit_transform(df_standardized2)
print(pca2.explained_variance_ratio_)
```

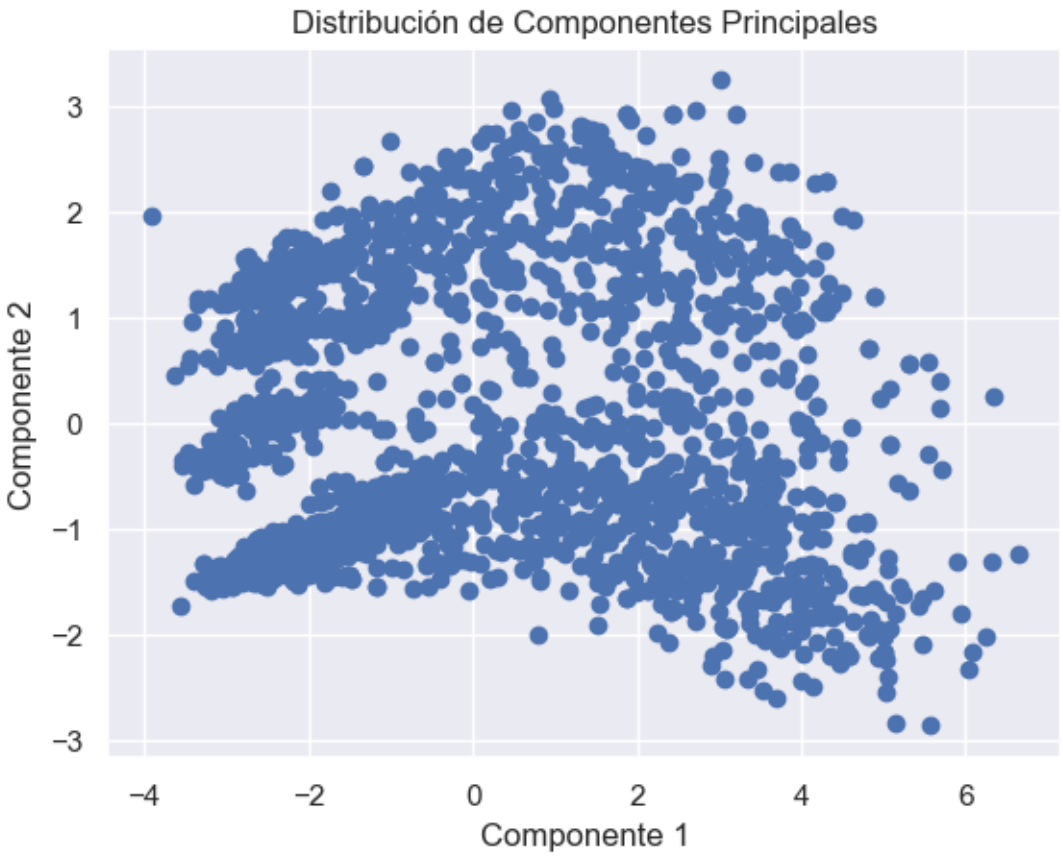
```
[0.33480884 0.09751151 0.0782904 ]
```

[152]:
```
pca_df2 = pd.DataFrame(data=pca2_features,columns=['PC1', 'PC2', 'PC3'])
pca_df2.describe().apply(lambda s: s.apply('{0:.3f}'.format))
```

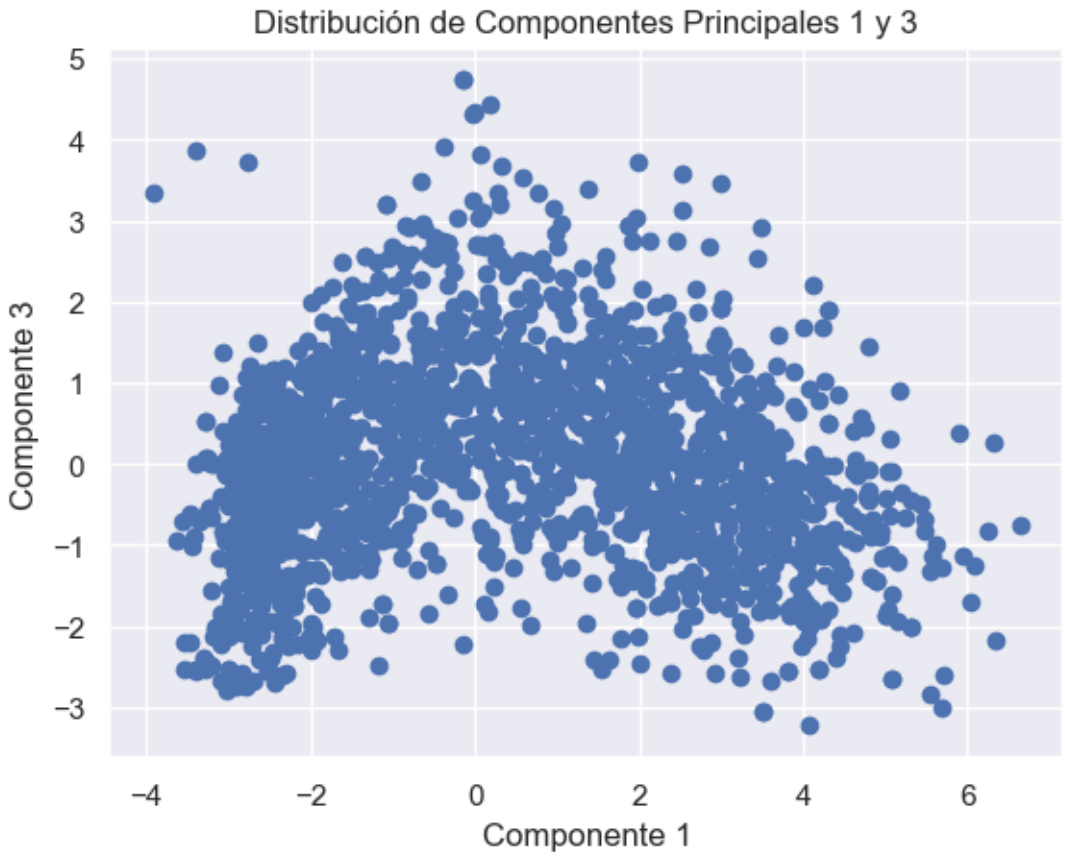
[152]:

| | PC1 | PC2 | PC3 |
|---|---|---|---|
| count | 2205.000 | 2205.000 | 2205.000 |
| mean | 0.000 | -0.000 | -0.000 |
| std | 2.455 | 1.325 | 1.187 |
| min | -3.919 | -2.867 | -3.223 |
| 25% | -2.340 | -1.226 | -0.775 |
| 50% | -0.502 | -0.307 | -0.059 |
| 75% | 2.126 | 1.229 | 0.725 |
| max | 6.636 | 3.244 | 4.733 |

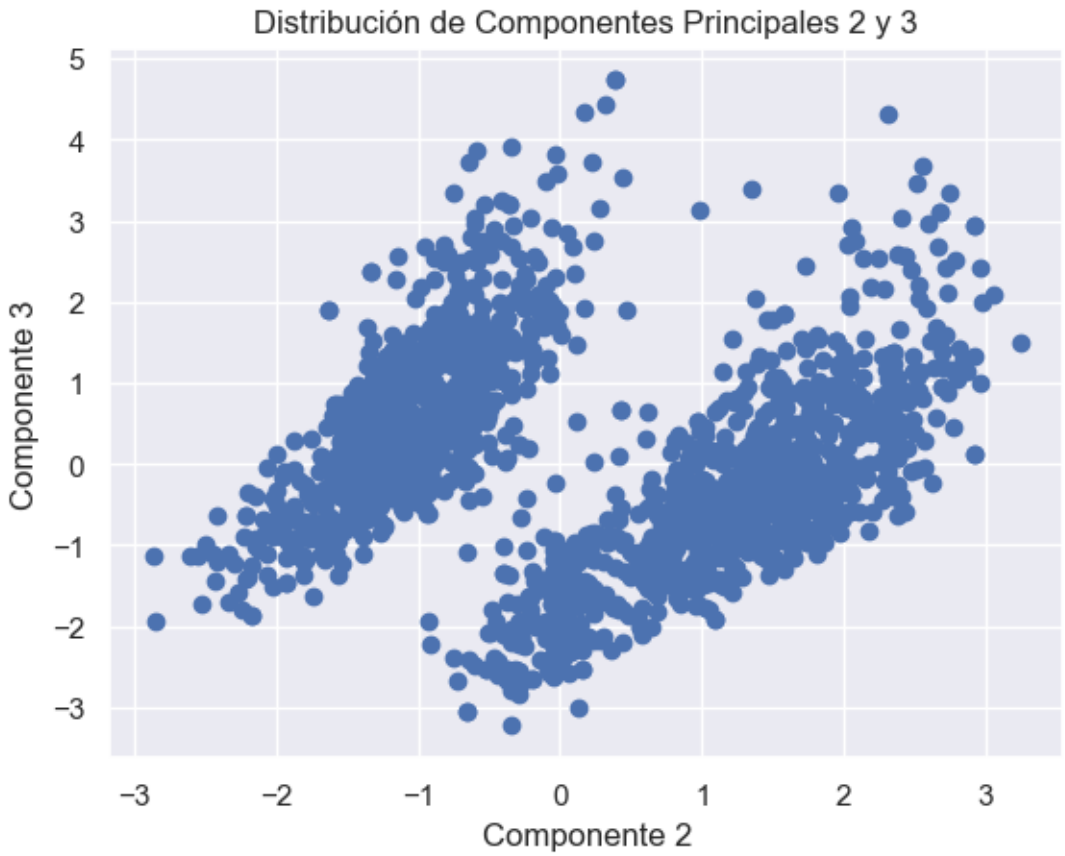
[153]:
```
plt.scatter(pca_df2['PC1'],pca_df2['PC2'])
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.title('Distribución de Componentes Principales 1 y 2')
plt.show()
```



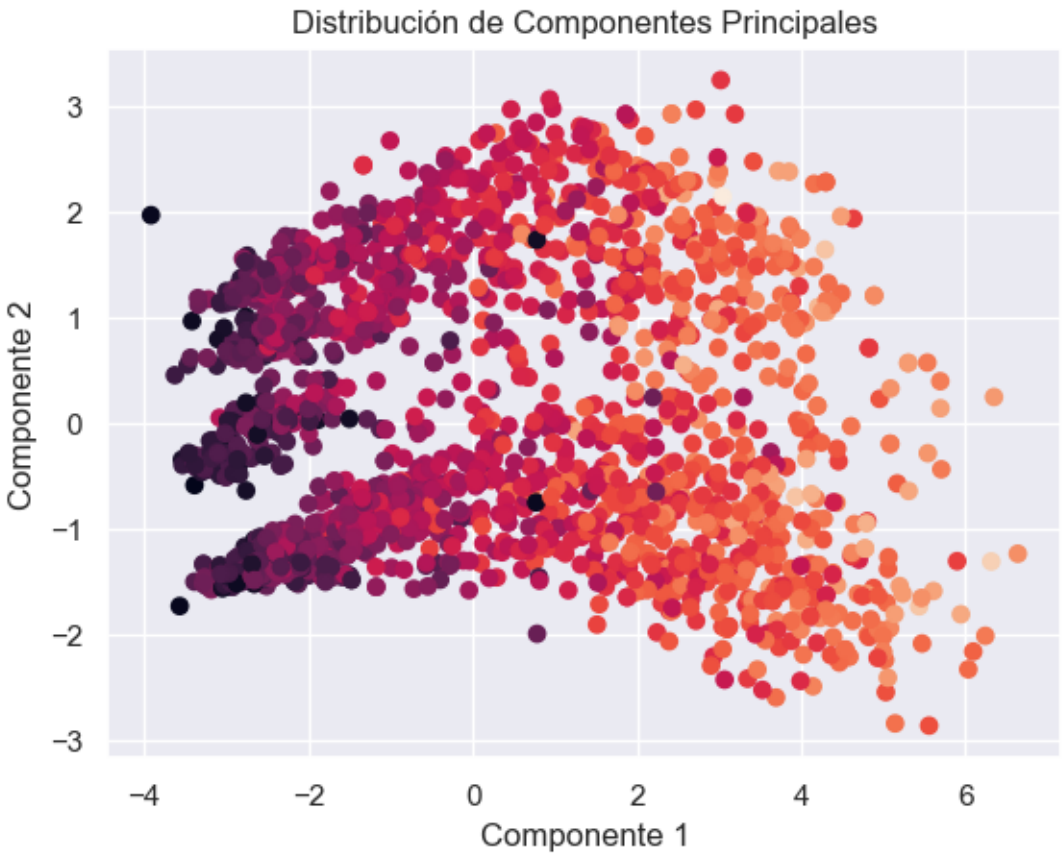

[184]:
```
plt.scatter(pca_df2['PC1'],pca_df2['PC3'])
plt.xlabel('Componente 1')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales 1 y 3')
plt.show()
```



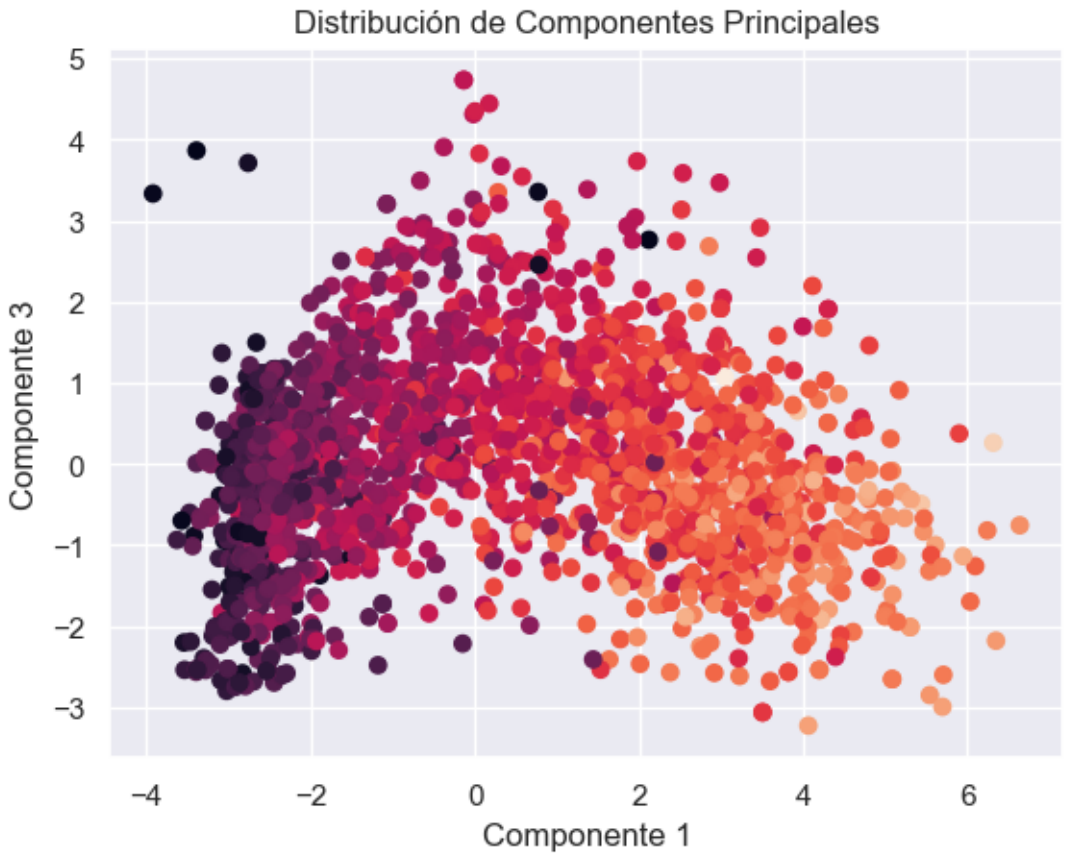

[185]:
```
plt.scatter(pca_df2['PC2'],pca_df2['PC3'])
plt.xlabel('Componente 2')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales 2 y 3')
plt.show()
```



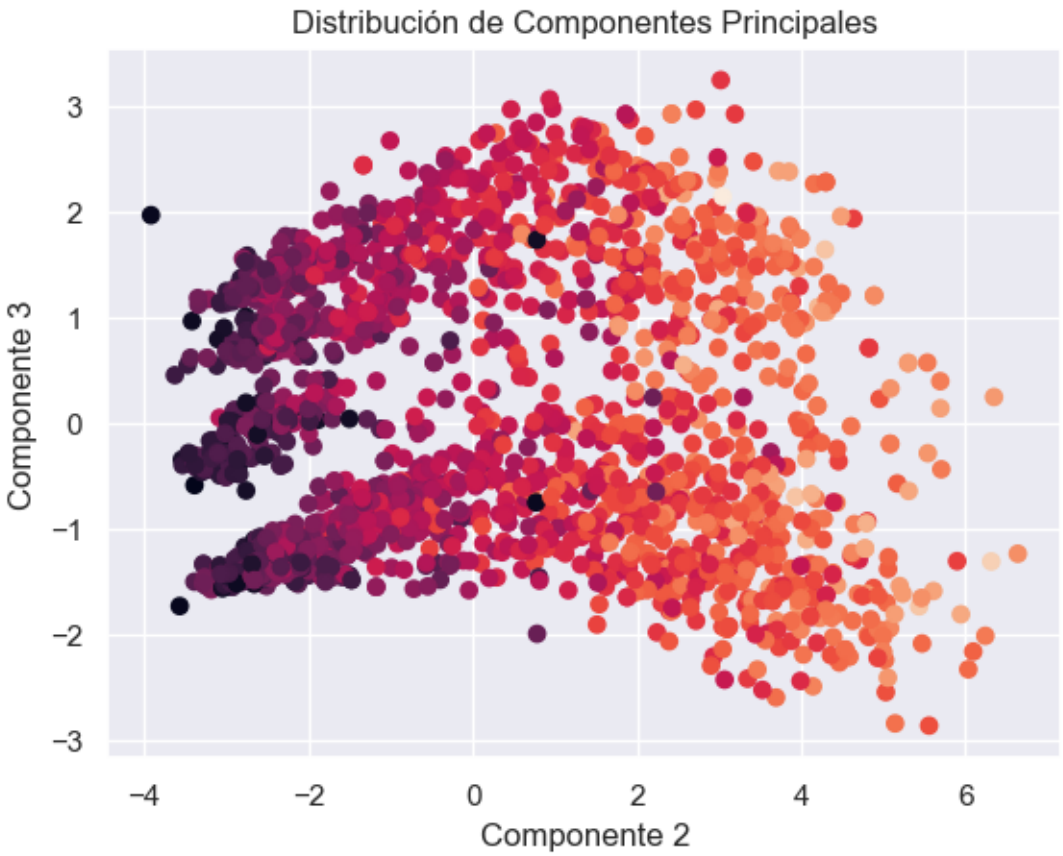

[186]:
```
# Obtener las componentes PC1 y PC2 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC2']
# Obtener la variable "Income" de la base de datos df_standardized2
income = df_standardized2['Income']
# Generar colores aleatorios
colors = np.random.rand(len(x))
# Crear el gráfico de dispersión
plt.scatter(x, y, c=income)
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```




[194]:
```
# Obtener las componentes PC1 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
income = df_standardized2['Income']
# Generar colores aleatorios
colors = np.random.rand(len(x))
# Crear el gráfico de dispersión
plt.scatter(x, y, c=income)
plt.xlabel('Componente 1')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

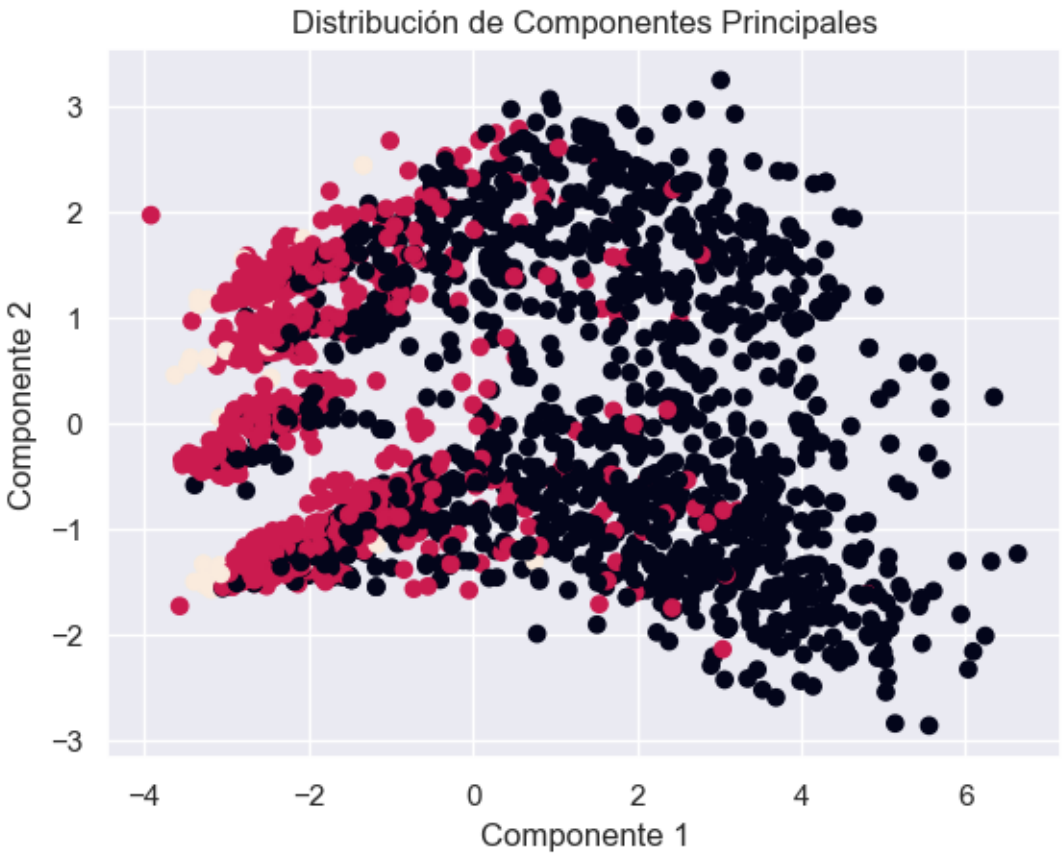



[195]:
```
# Obtener las componentes PC2 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC2']
# Obtener la variable "Income" de la base de datos df_standardized2
income = df_standardized2['Income']
# Generar colores aleatorios
colors = np.random.rand(len(x))
# Crear el gráfico de dispersión
plt.scatter(x, y, c=income)
plt.xlabel('Componente 2')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

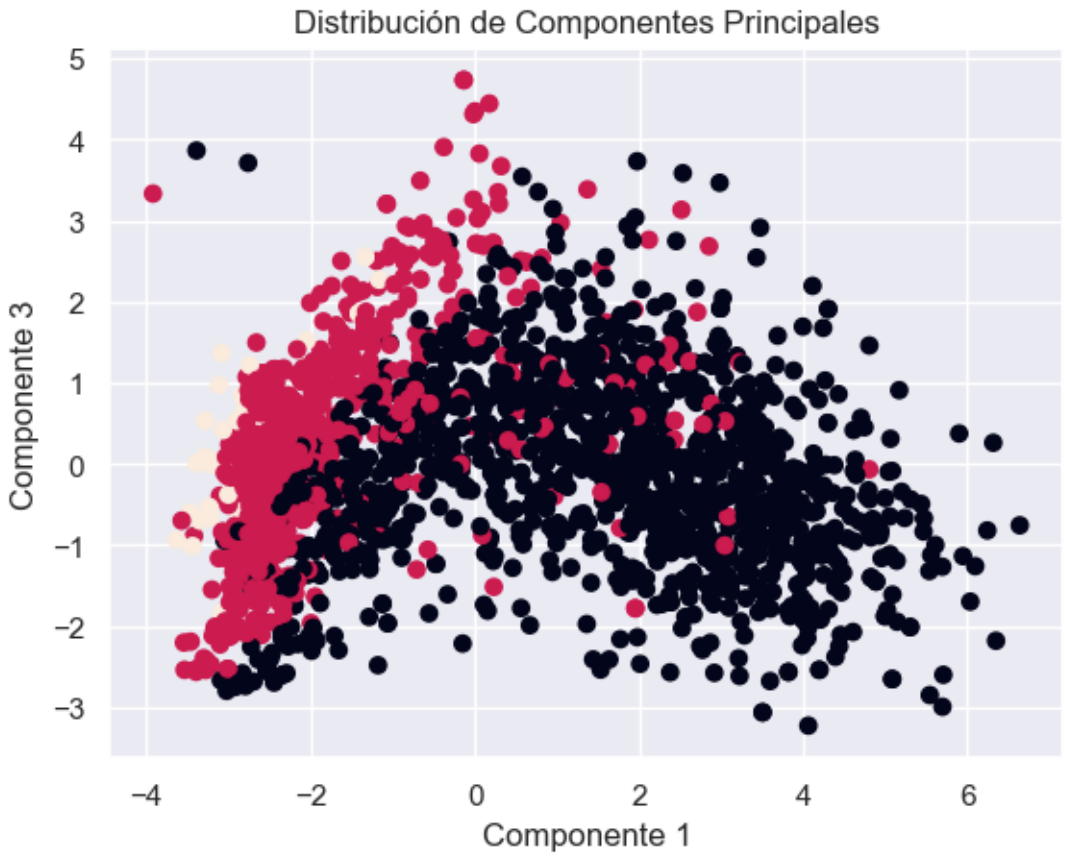



En este caso, se observa que la componente 1 y la componente 2 son las que explican la variabilidad del ingreso.

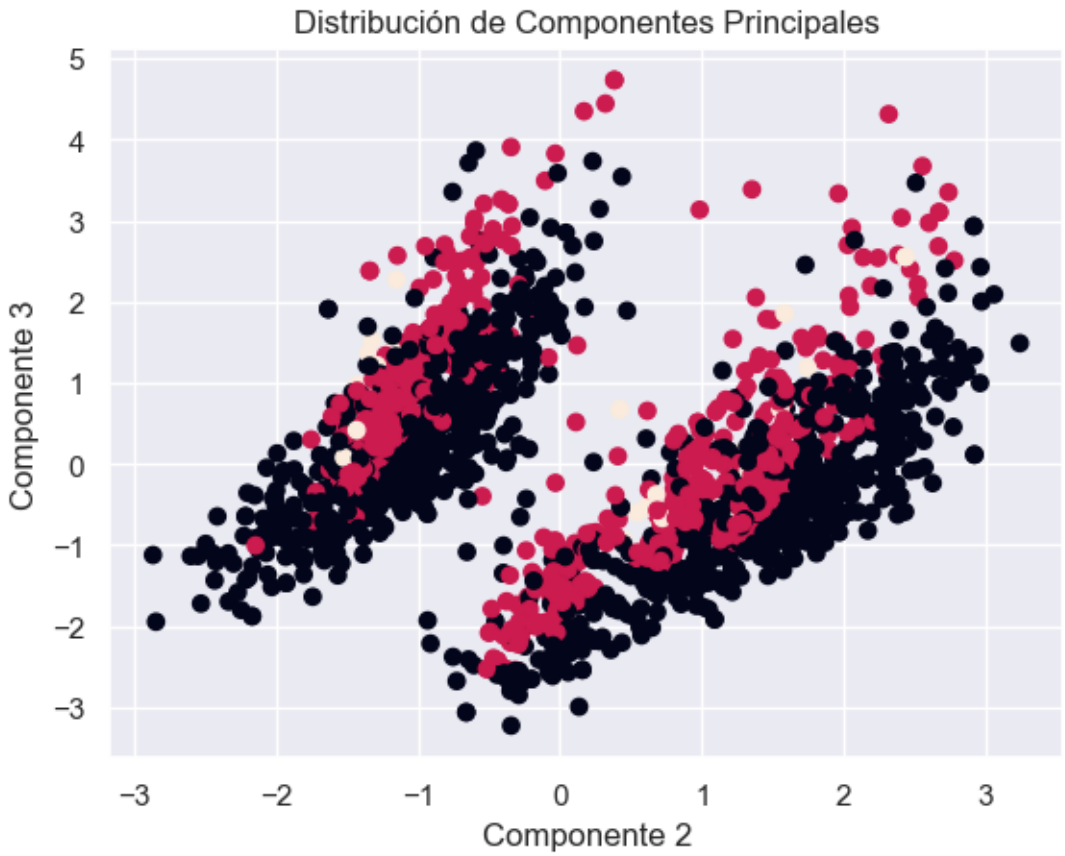
[187]:
```
# Obtener las componentes PC1 y PC2 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC2']
# Obtener la variable "Income" de la base de datos df_standardized2
Kidhome = df_standardized2['Kidhome']
# Generar colores aleatorios
colors = np.random.rand(len(x))
# Crear el gráfico de dispersión
plt.scatter(x, y, c=Kidhome)
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```




[196]:
```
# Obtener las componentes PC1 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
Kidhome = df_standardized2['Kidhome']
# Generar colores aleatorios
colors = np.random.rand(len(x))
# Crear el gráfico de dispersión
plt.scatter(x, y, c=Kidhome)
plt.xlabel('Componente 1')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

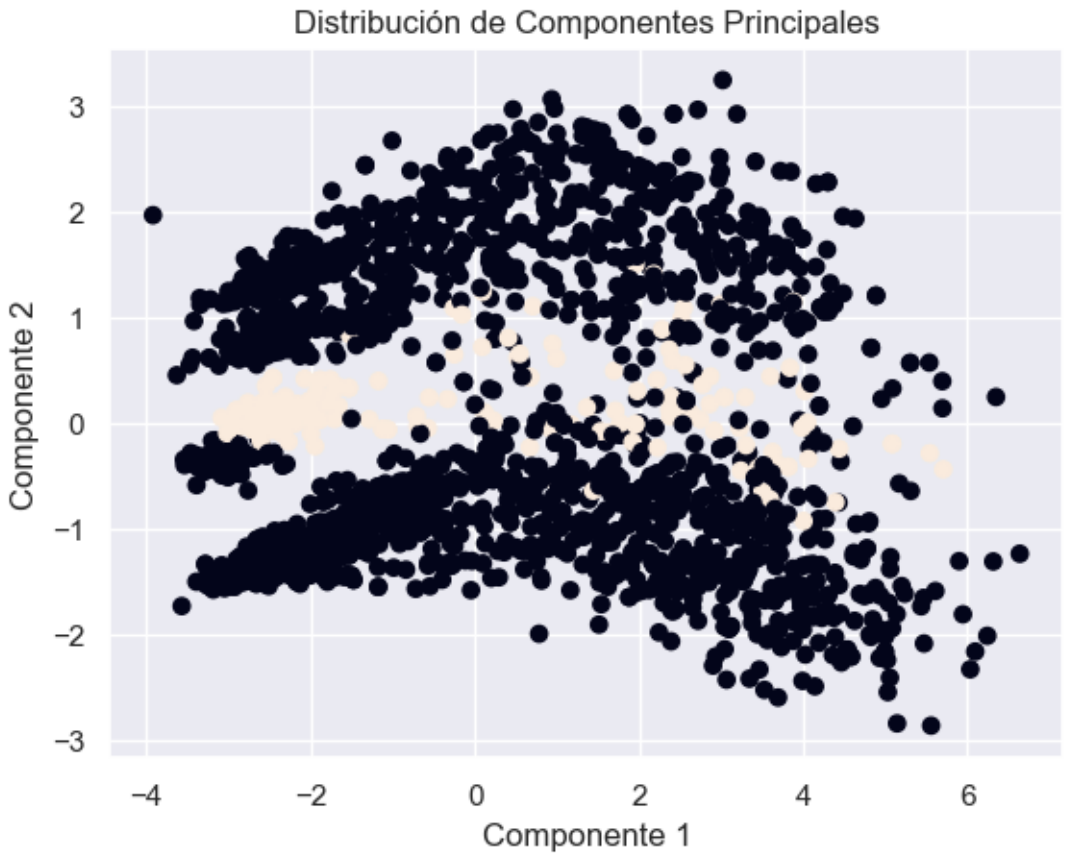



[199]:
```
# Obtener las componentes PC2 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC2']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
Kidhome = df_standardized2['Kidhome']
# Generar colores aleatorios
colors = np.random.rand(len(x))
# Crear el gráfico de dispersión
plt.scatter(x, y, c=Kidhome)
plt.xlabel('Componente 2')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

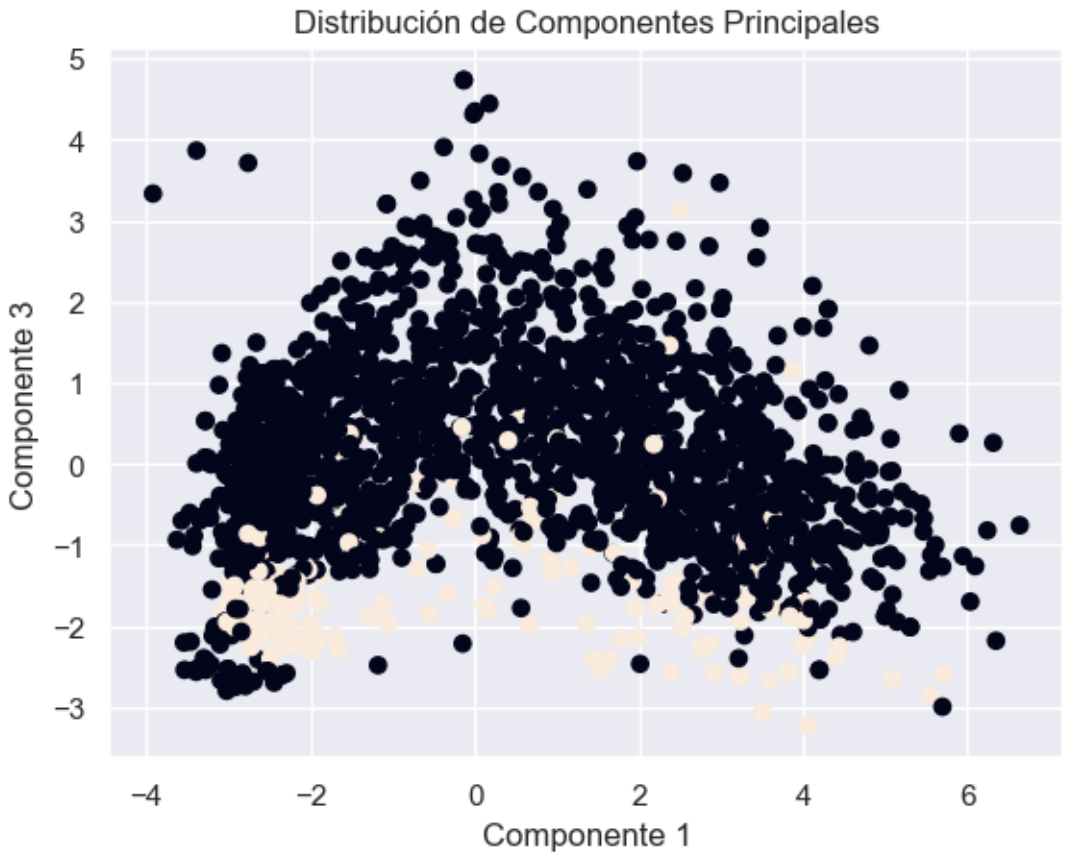



Aquí se observa que todas las componentes explican la variabilidad de si hay niños en casa, ya que todas segregan de alguna forma los valores de la variable.

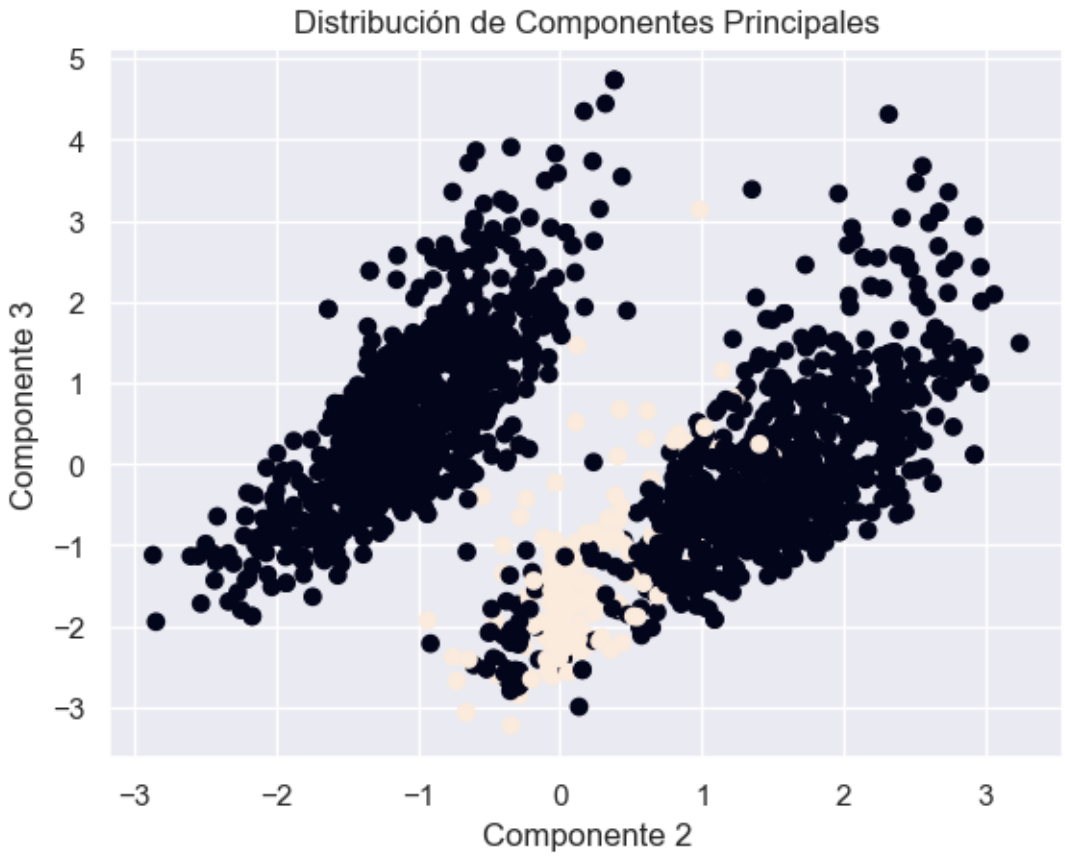
[188]:
```
# Obtener las componentes PC1 y PC2 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC2']
# Obtener la variable "Income" de la base de datos df_standardized2
e1 = df_standardized2['education_2n Cycle']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e1)
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```




[198]:
```
# Obtener las componentes PC1 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
e1 = df_standardized2['education_2n Cycle']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e1)
plt.xlabel('Componente 1')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

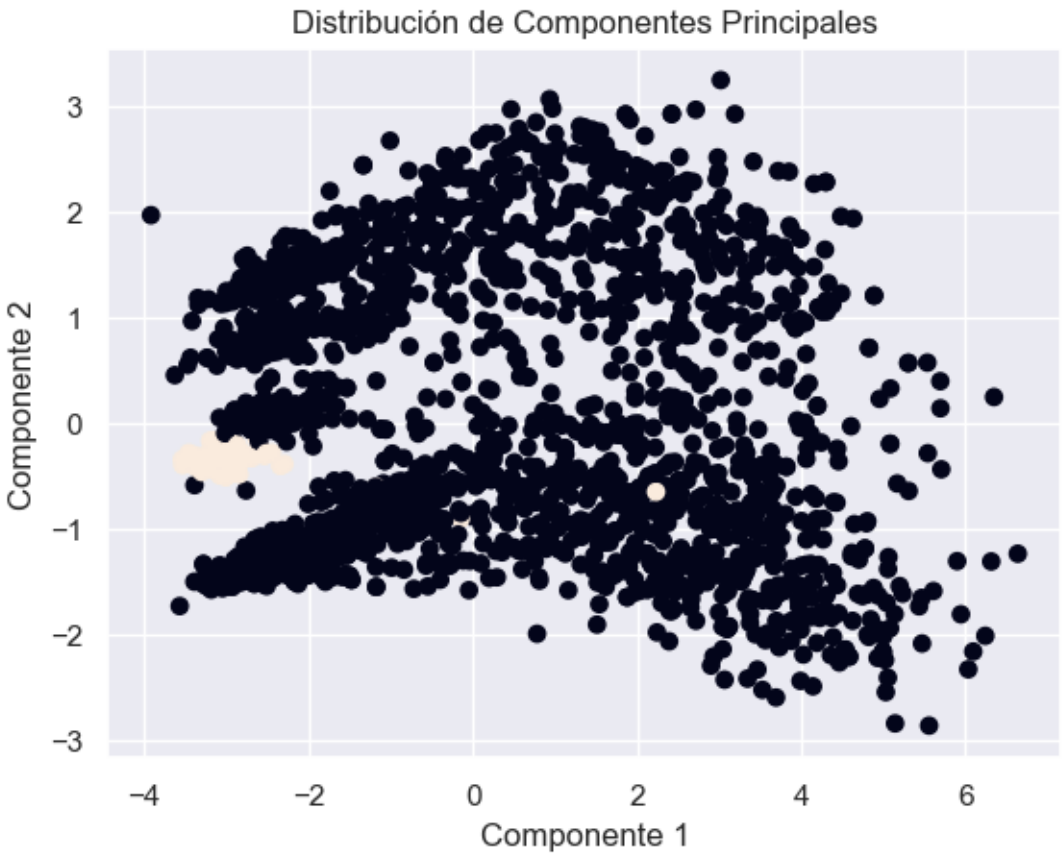



[200]:
```
# Obtener las componentes PC2 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC2']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
e1 = df_standardized2['education_2n Cycle']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e1)
plt.xlabel('Componente 2')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

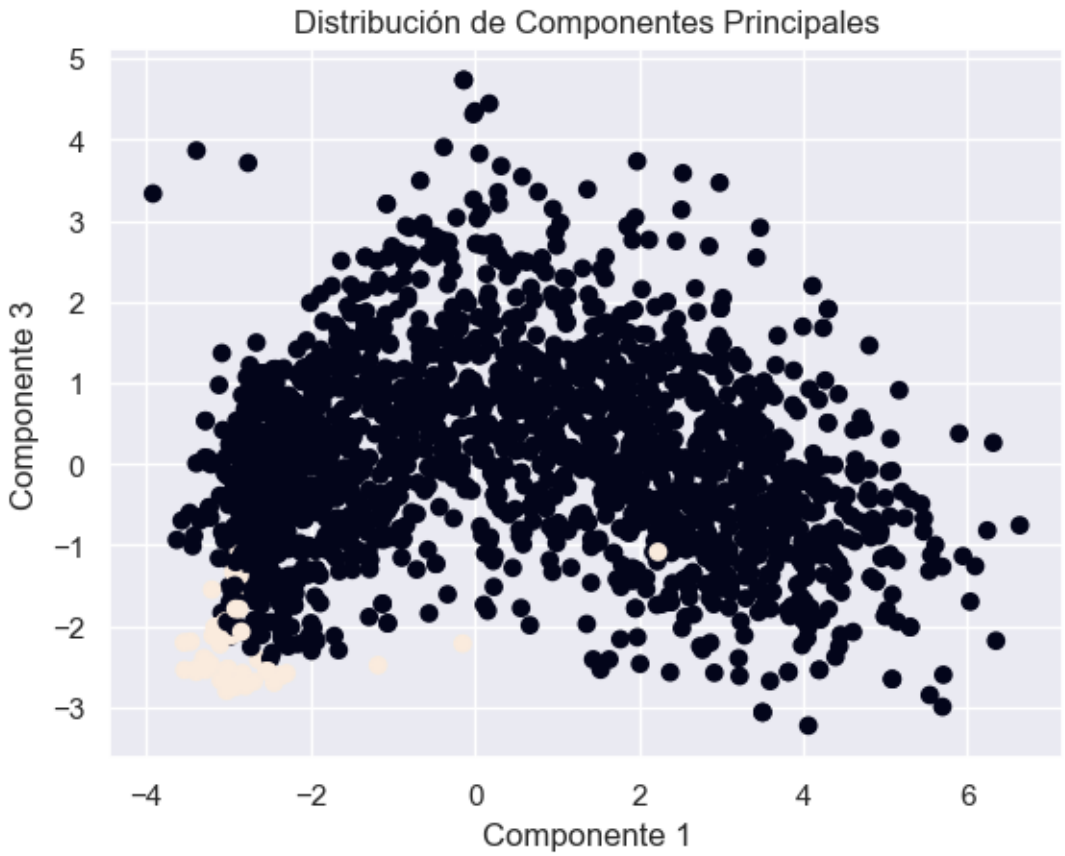



Cuando la componente 2 toma valores cercanos a 0, se observa que ocurre una separación de grupos entre los valores que la variable de si la persona completo su secundaria. Luego, es posible afirmar que la componente 2 explica la variabilidad de la variable mencionada, pero no de forma tan clara.

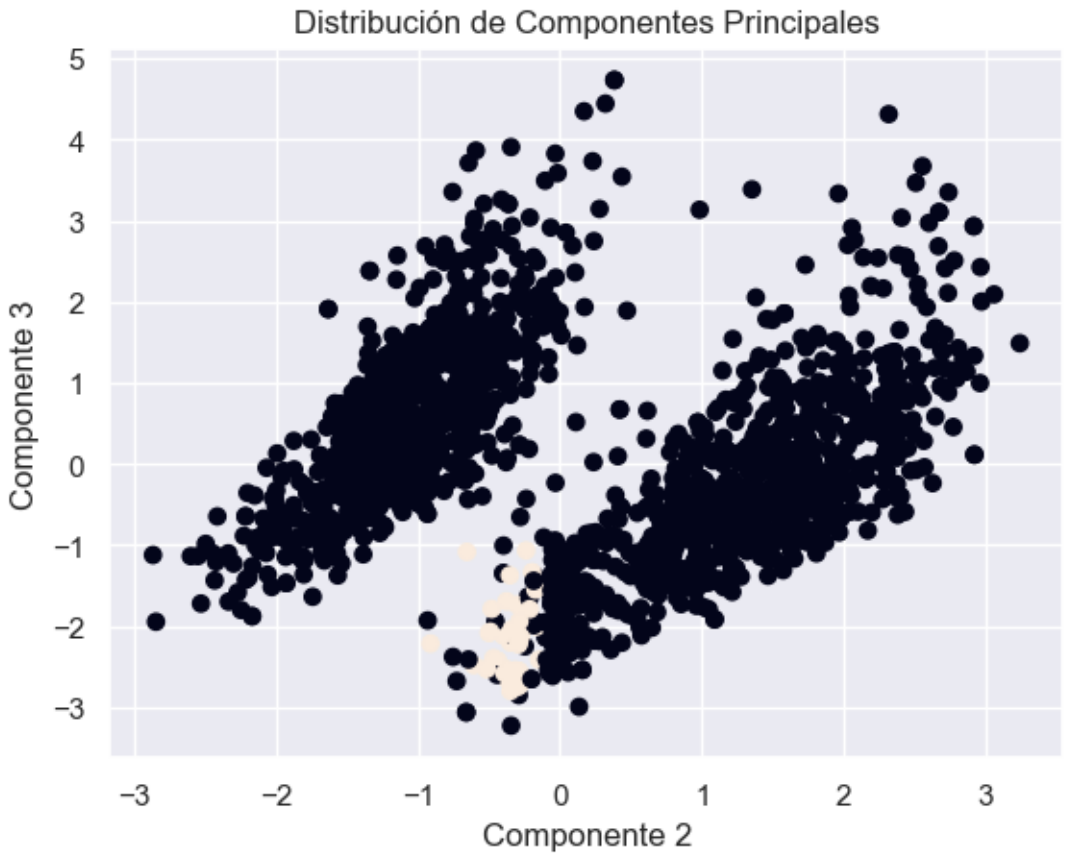
[189]:
```
# Obtener las componentes PC1 y PC2 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC2']
# Obtener la variable "Income" de la base de datos df_standardized2
e2 = df_standardized2['education_Basic']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e2)
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```




[201]:
```
# Obtener las componentes PC1 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
e2 = df_standardized2['education_Basic']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e2)
plt.xlabel('Componente 1')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

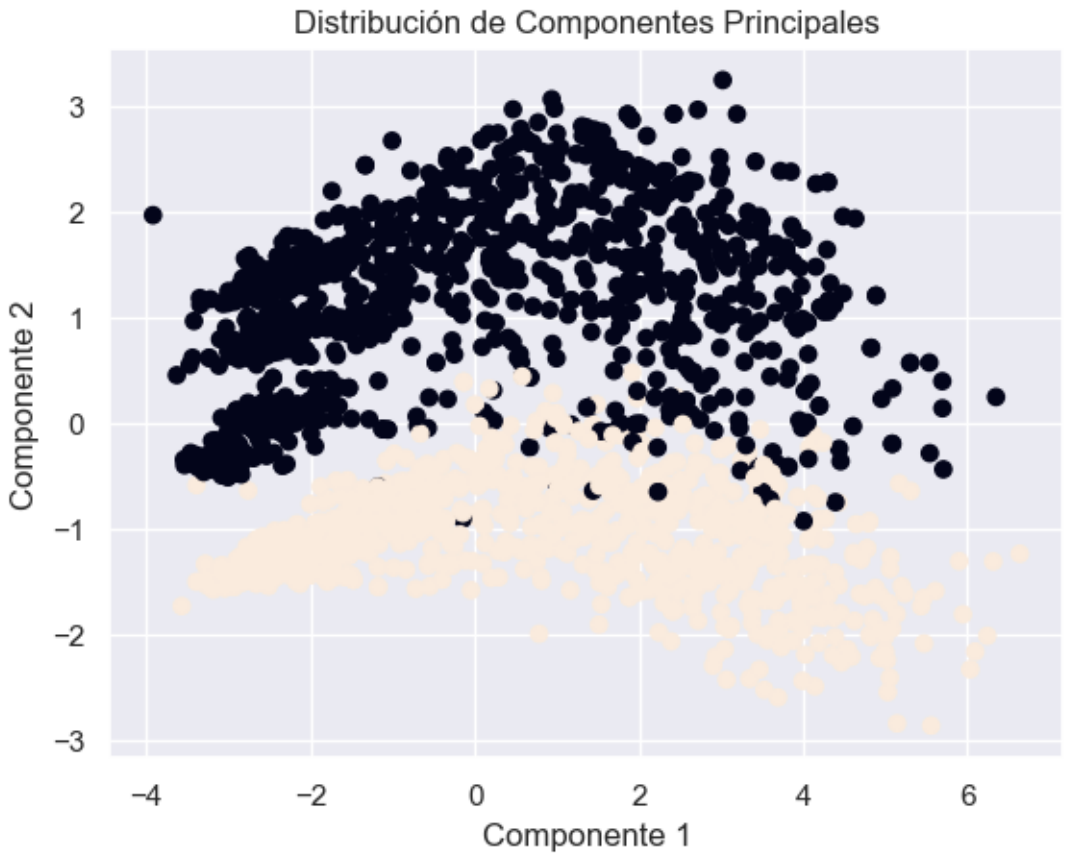



[202]:
```
# Obtener las componentes PC2 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC2']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
e2 = df_standardized2['education_Basic']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e2)
plt.xlabel('Componente 2')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

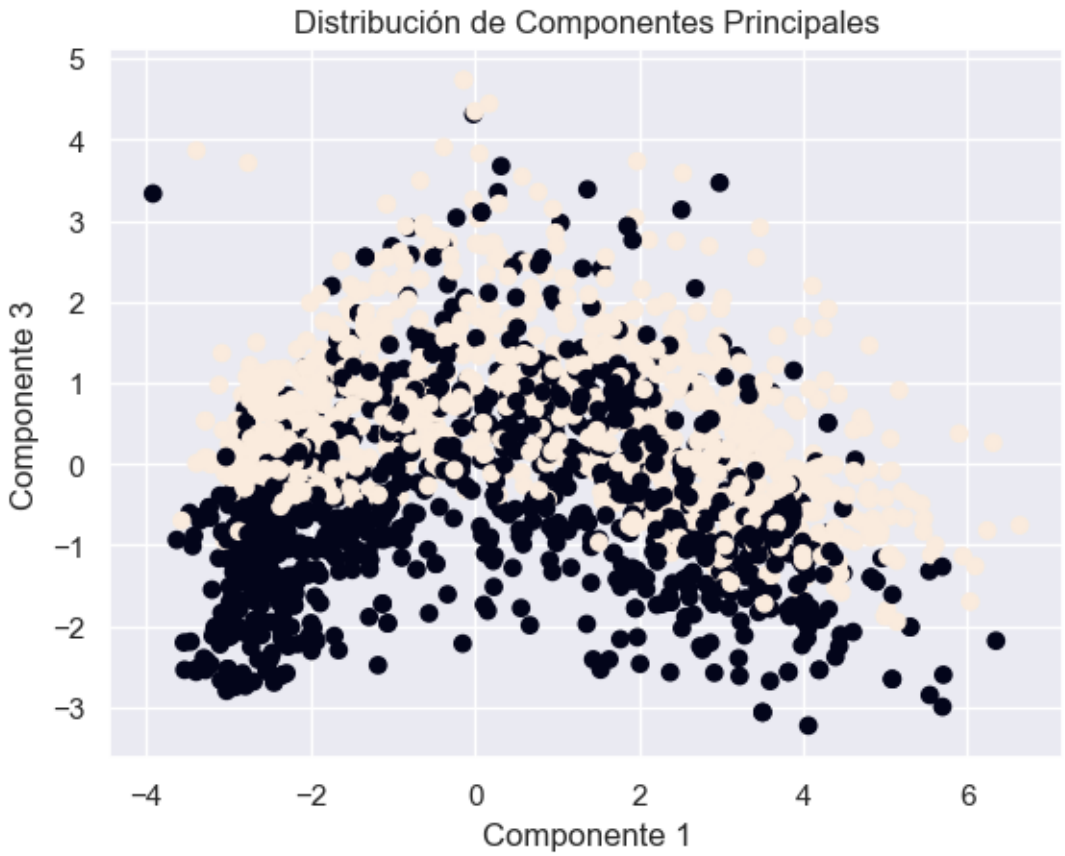



Se observa que no hay claridad en la formación de clusters. La diferencia entre los valores de las observaciones dificulta la interpretación de la relación entre los componentes.

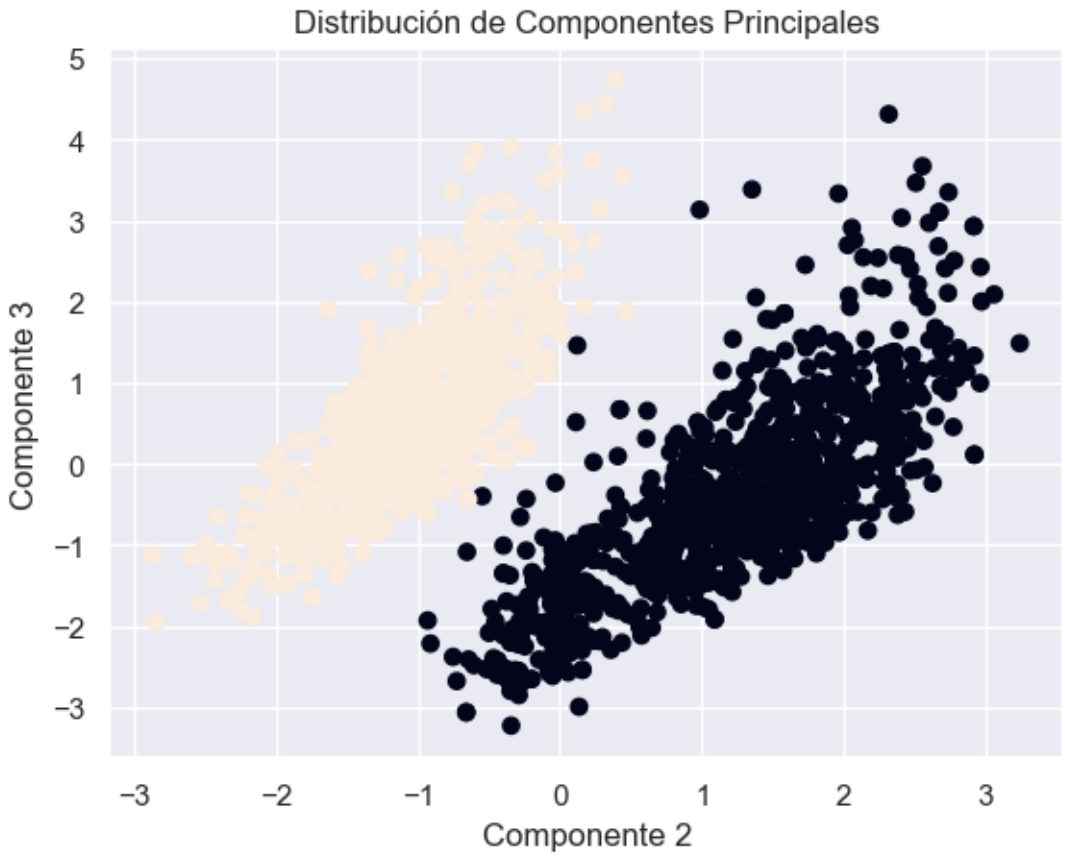
[190]:
```
# Obtener las componentes PC1 y PC2 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC2']
# Obtener la variable "Income" de la base de datos df_standardized2
e3 = df_standardized2['education_Graduation']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e3)
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```




[203]:
```
# Obtener las componentes PC1 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
e3 = df_standardized2['education_Graduation']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e3)
plt.xlabel('Componente 1')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

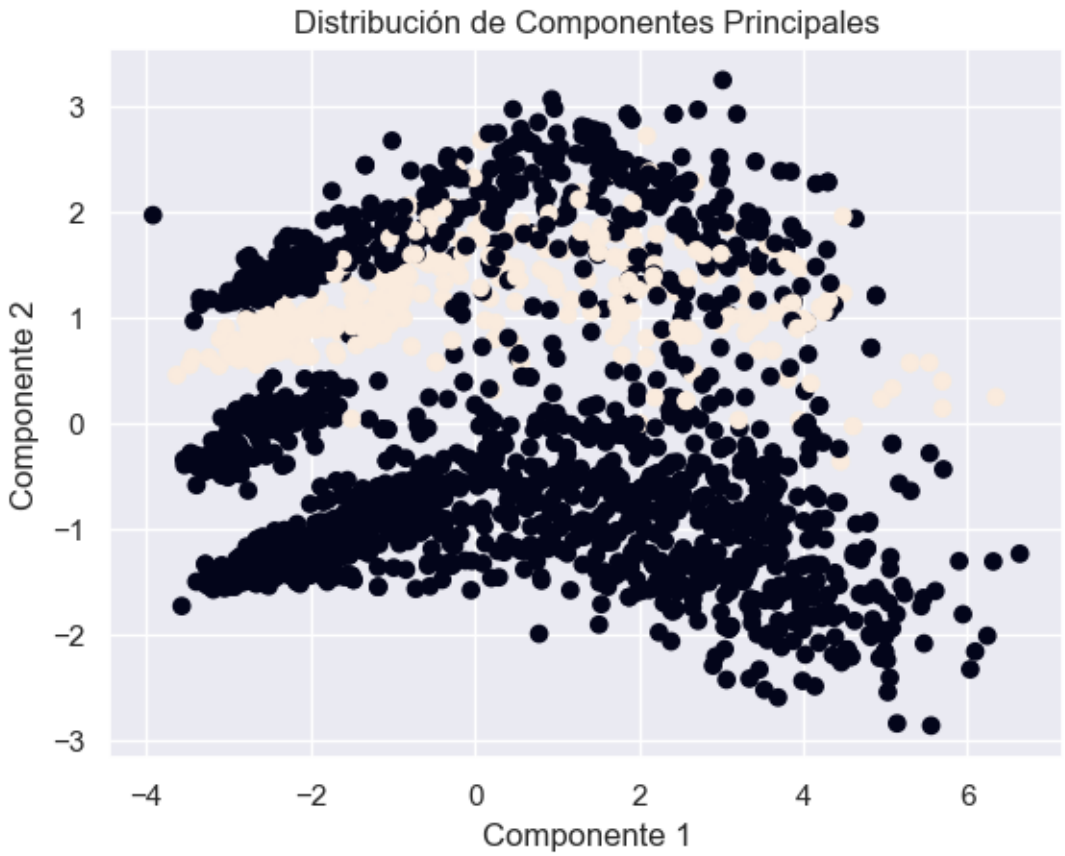



[204]:
```
# Obtener las componentes PC2 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC2']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
e3 = df_standardized2['education_Graduation']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e3)
plt.xlabel('Componente 2')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

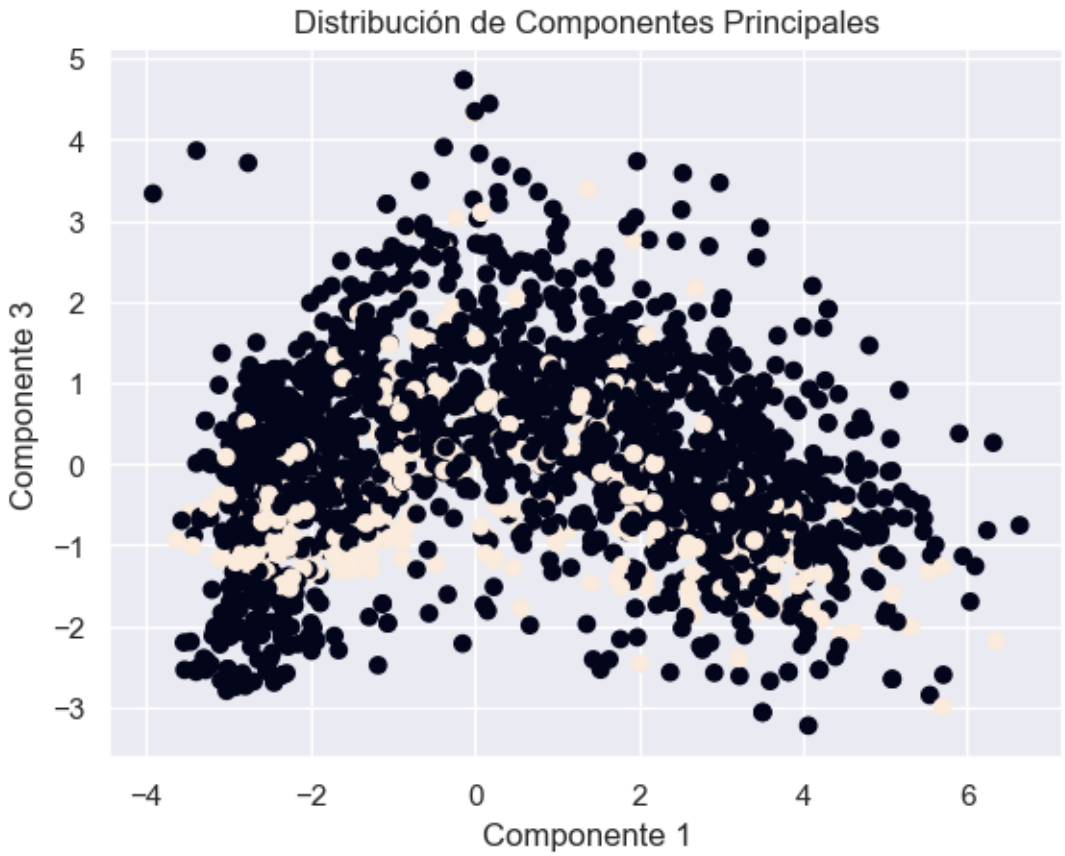



Se ve claramente que la componente 2 explica la variabilidad de la variable de si la persona completo su estudio universitario, ya que esta separa los grupos en clusters cuando cambia de signo.

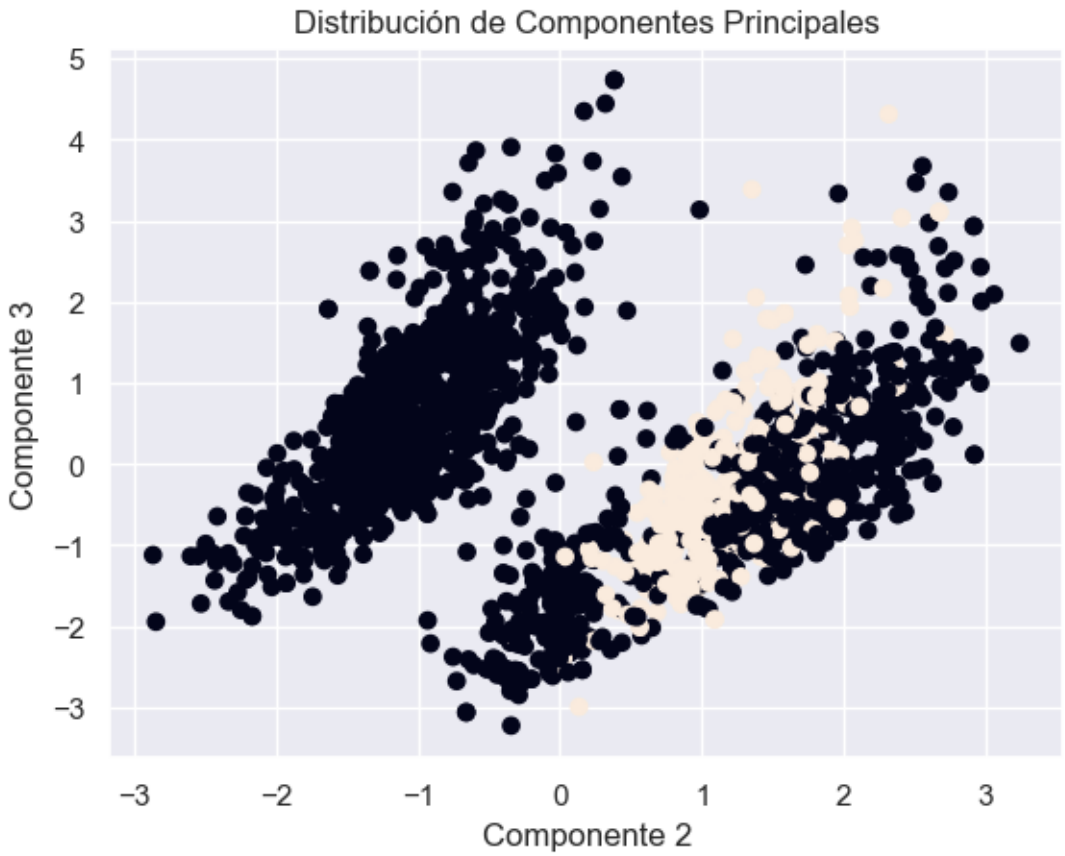
[191]:
```
# Obtener las componentes PC1 y PC2 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC2']
# Obtener la variable "Income" de la base de datos df_standardized2
e4 = df_standardized2['education_Master']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e4)
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```




[205]:
```
# Obtener las componentes PC1 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
e4 = df_standardized2['education_Master']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e4)
plt.xlabel('Componente 1')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

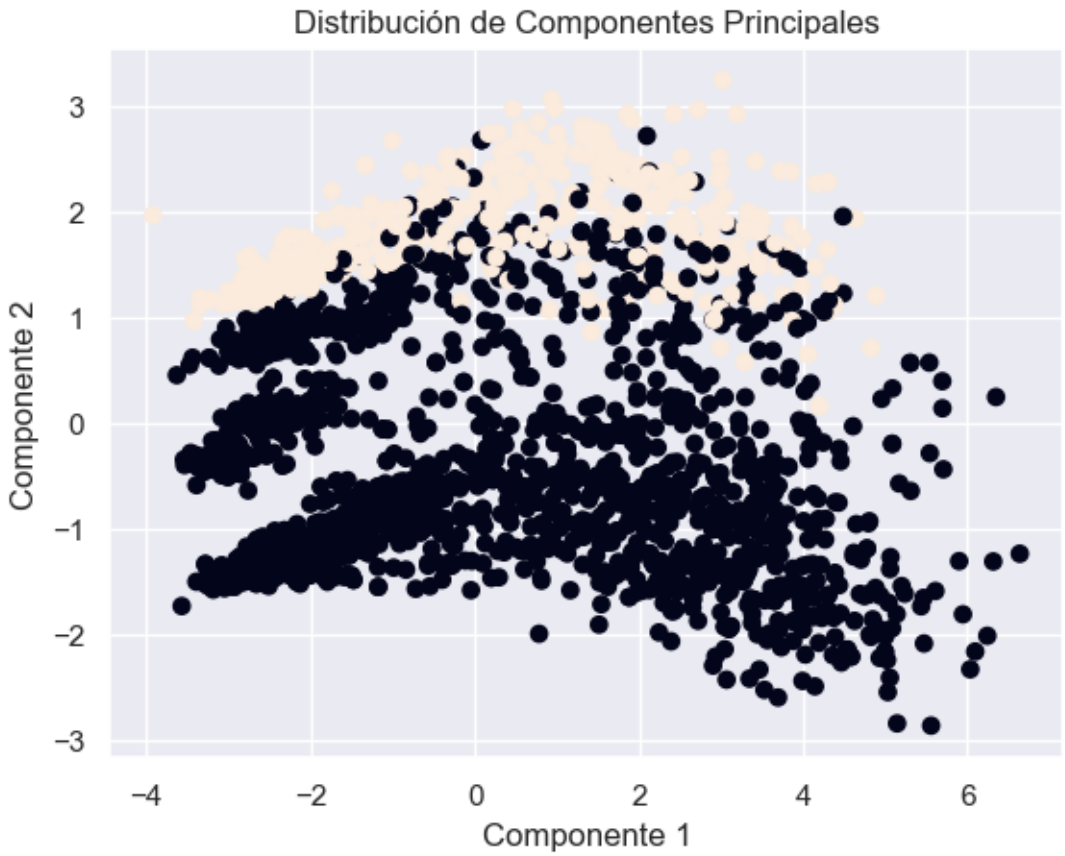



[206]:
```
# Obtener las componentes PC2 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC2']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
e4 = df_standardized2['education_Master']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e4)
plt.xlabel('Componente 2')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

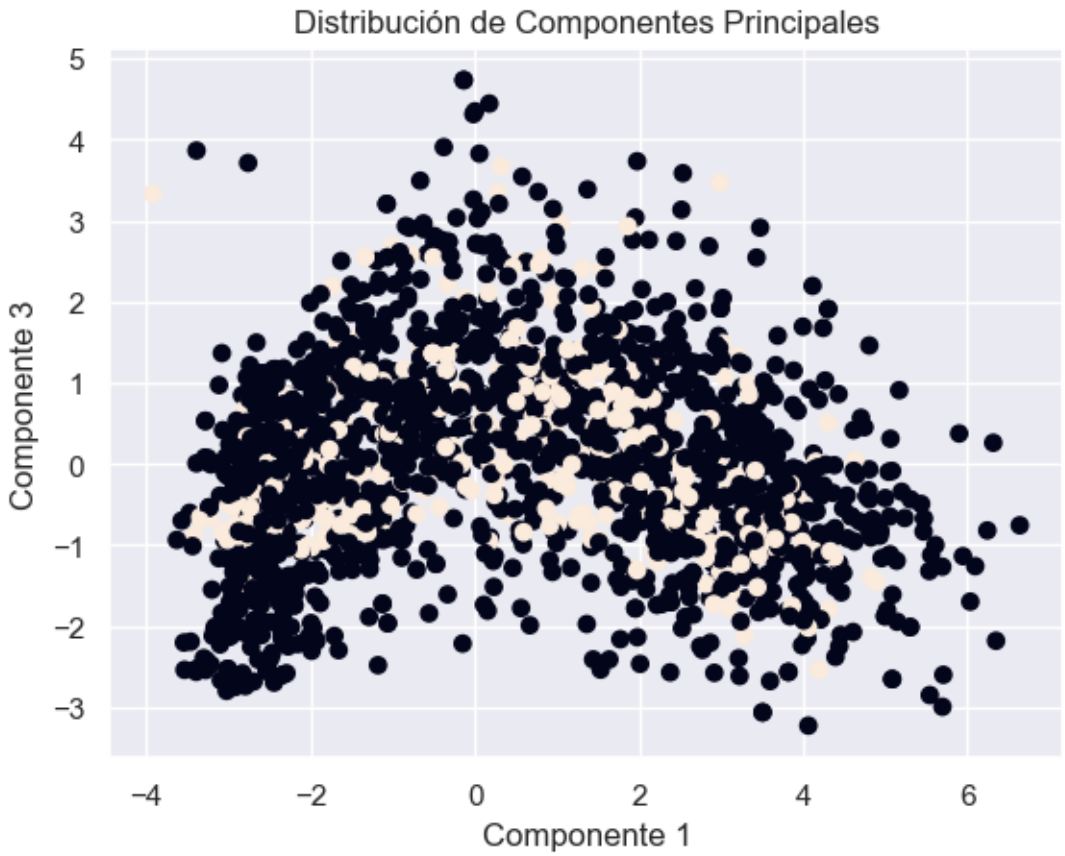



No se observa claridad de formación de clusters. Sin embargo, cuando la componente es positiva entre 0 y 1, se observa una discontinuidad de cierto grupo de valores que toma la variable de si la persona tiene magister.

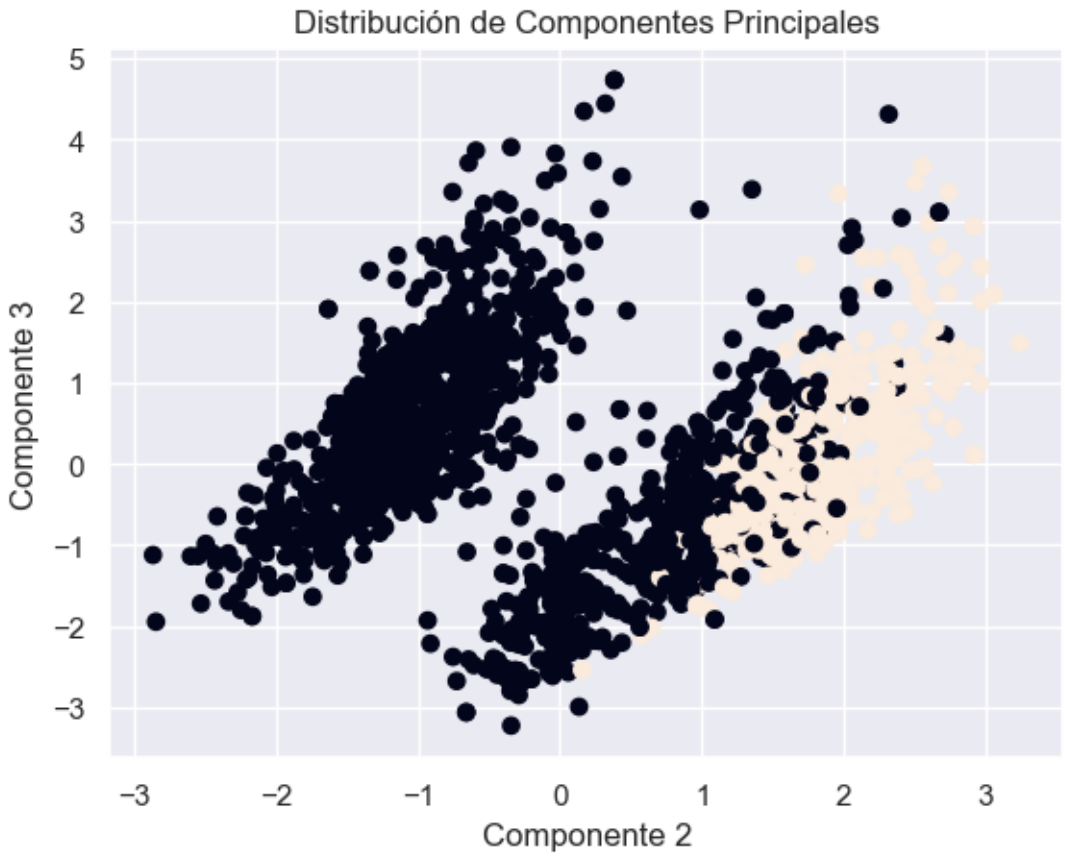
[192]:
```
# Obtener las componentes PC1 y PC2 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC2']
# Obtener la variable "Income" de la base de datos df_standardized2
e5 = df_standardized2['education_PhD']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e5)
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```




[207]:
```
# Obtener las componentes PC1 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
e5 = df_standardized2['education_PhD']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e5)
plt.xlabel('Componente 1')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

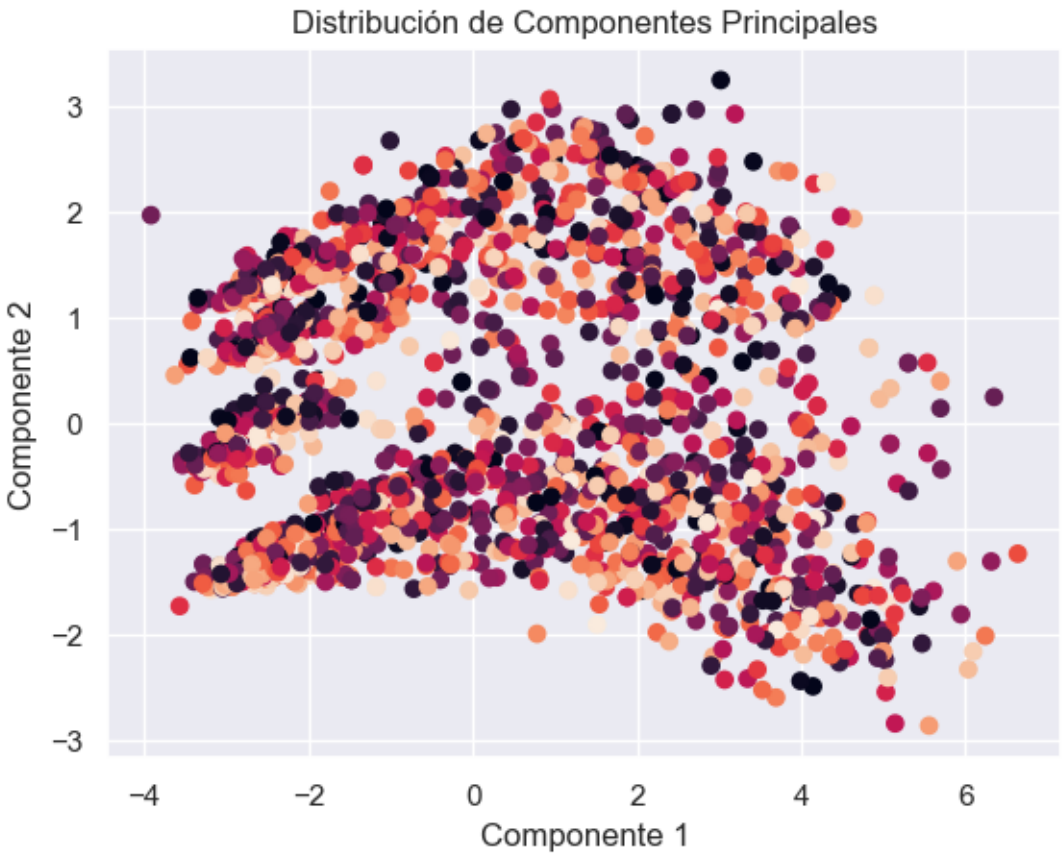



[208]:
```
# Obtener las componentes PC2 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC2']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
e5 = df_standardized2['education_PhD']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=e5)
plt.xlabel('Componente 2')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```

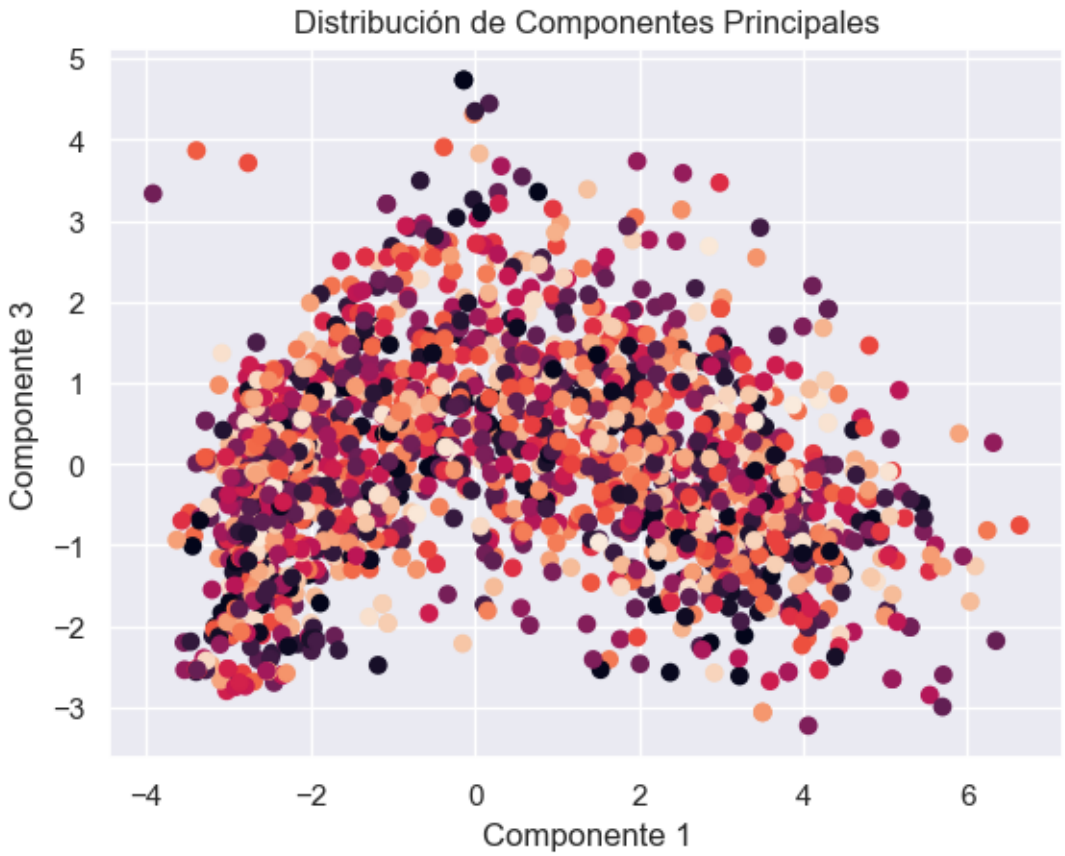



Se observa que no hay claridad en la formación de clusters. Sin embargo, se observa que la componente 2 es la que más separa a los valores de si la persona tiene doctorado, tomando cierto valor cuando esta es positiva y otro cuando es negativa.

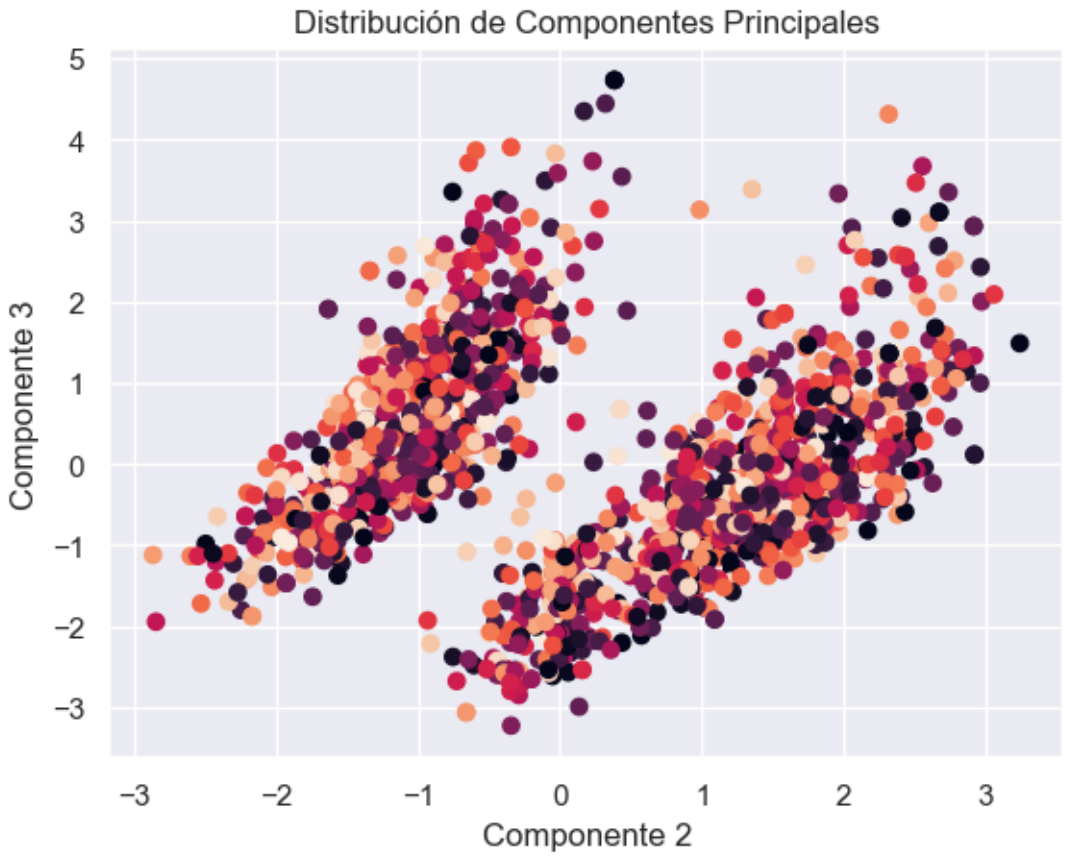
[193]:
```
# Obtener las componentes PC1 y PC2 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC2']
# Obtener la variable "Income" de la base de datos df_standardized2
Recency = df_standardized2['Recency']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=Recency)
plt.xlabel('Componente 1')
plt.ylabel('Componente 2')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```




[209]:
```
# Obtener las componentes PC1 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC1']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
Recency = df_standardized2['Recency']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=Recency)
plt.xlabel('Componente 1')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```




[210]:
```
# Obtener las componentes PC2 y PC3 de la base de datos de componentes pca_df2
x = pca_df2['PC2']
y = pca_df2['PC3']
# Obtener la variable "Income" de la base de datos df_standardized2
Recency = df_standardized2['Recency']
# Crear el gráfico de dispersión
plt.scatter(x, y, c=Recency)
plt.xlabel('Componente 2')
plt.ylabel('Componente 3')
plt.title('Distribución de Componentes Principales')
# Mostrar el gráfico
plt.show()
```




No existe una clasificación. No se observa separación de grupos. Se concluye que ninguno de estos componentes explica la variable Recency.

### 1.3.1 Pregunta 4

## 1.4 EFA

A continuación, se realizará un análisis factorial exploratorio para identificar factores que estén asociados a un grupo de variables.

[83]:
```
from factor_analyzer import FactorAnalyzer

# Selección de las variables deseadas
variables = ['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts',␣
 ↪'MntSweetProducts', 'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',␣
 ↪'NumCatalogPurchases', 'NumStorePurchases']

# Subset del dataframe con las variables seleccionadas
df_subset = df_food[variables]

# Creación del objeto FactorAnalyzer con rotación promax
```

```
fa = FactorAnalyzer(rotation='promax')

# Realizar el análisis factorial en el subset de datos
fa.fit(df_subset)
```

[83]:
```
FactorAnalyzer(rotation_kwargs={})
```

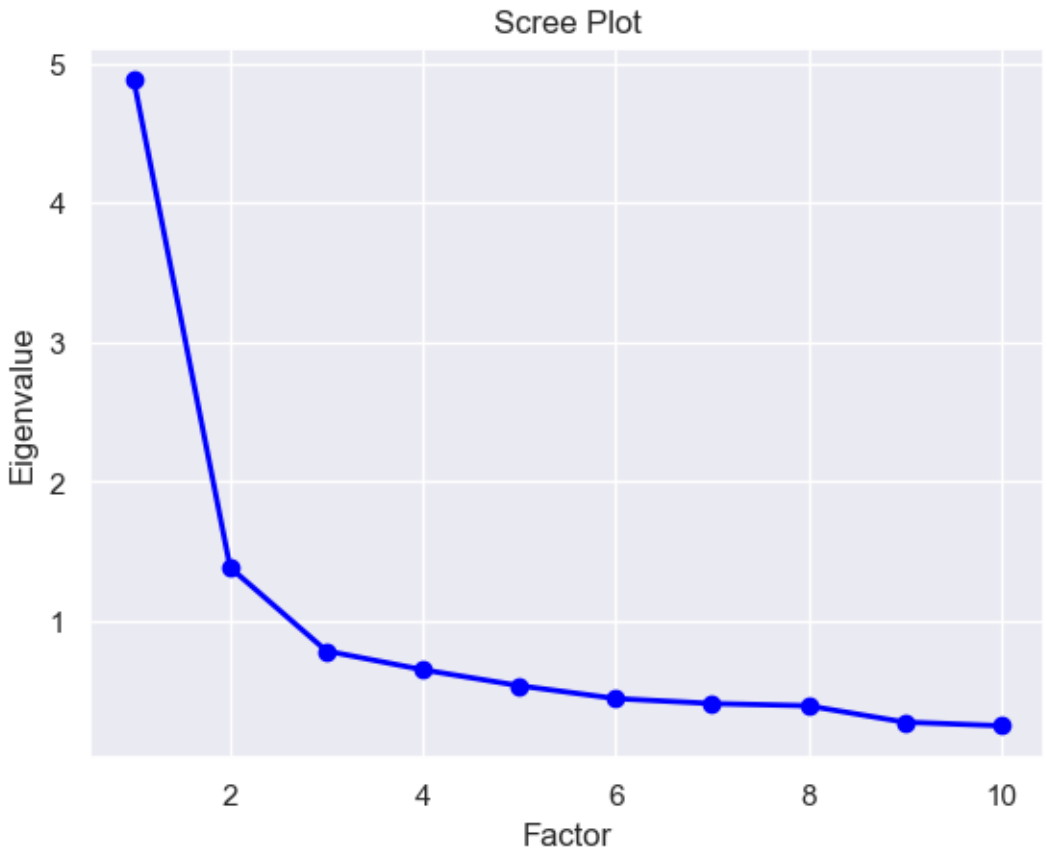
[84]:
```
fa.loadings_
```

[84]:
```
array([[-0.26584335,  1.09735943,  0.00289643],
       [ 0.82308031, -0.08105236, -0.06126105],
       [ 0.38616794,  0.53194758, -0.27483943],
       [ 0.85317646, -0.07346727, -0.08608688],
       [ 0.77468459, -0.0445247 , -0.04279369],
       [ 0.46120536,  0.07316814,  0.2063711 ],
       [-0.13103401, -0.062116  ,  0.52848041],
       [ 0.19107856,  0.28559818,  0.55333599],
       [ 0.27834438,  0.62183945, -0.10832496],
       [ 0.25013369,  0.46755576,  0.17322481]])
```

[85]:
```
fa.get_eigenvalues()
```

[85]:
```
(array([4.87733524, 1.38320593, 0.78504073, 0.65035292, 0.53610436,
        0.44403425, 0.40825094, 0.39215129, 0.27476158, 0.24876277]),
 array([ 4.4999003 ,  0.97458021,  0.51150365,  0.1278122 ,  0.0757726 ,
         0.0412655 , -0.05612873, -0.10971925, -0.22163099, -0.25608689]))
```

[86]:
```
# Crear un array con los valores del 1 al 10 (correspondiente al número de␣
 ↪factores)
values = np.arange(1, 11)

# Obtener los valores propios del análisis factorial y guardarlos en un␣
 ↪DataFrame
eigenvalues = pd.DataFrame(data=fa.get_eigenvalues())

# Trazar el gráfico de Scree Plot
plt.plot(values, eigenvalues.iloc[0], 'o-', linewidth=2, color='blue')
plt.title('Scree Plot')
plt.xlabel('Factor')
plt.ylabel('Eigenvalue')
plt.show()
```




Se puede visualizar en el Scree Plot que a priori, las variables deberían agruparse en 2 factores, ya que el primer y el segundo factor tienen valores propios superiores a 1.

[87]:
```
fa.get_factor_variance()
```

[87]:
```
(array([2.6317396 , 2.19718631, 0.7583427 ]),
 array([0.26317396, 0.21971863, 0.07583427]),
 array([0.26317396, 0.48289259, 0.55872686]))
```

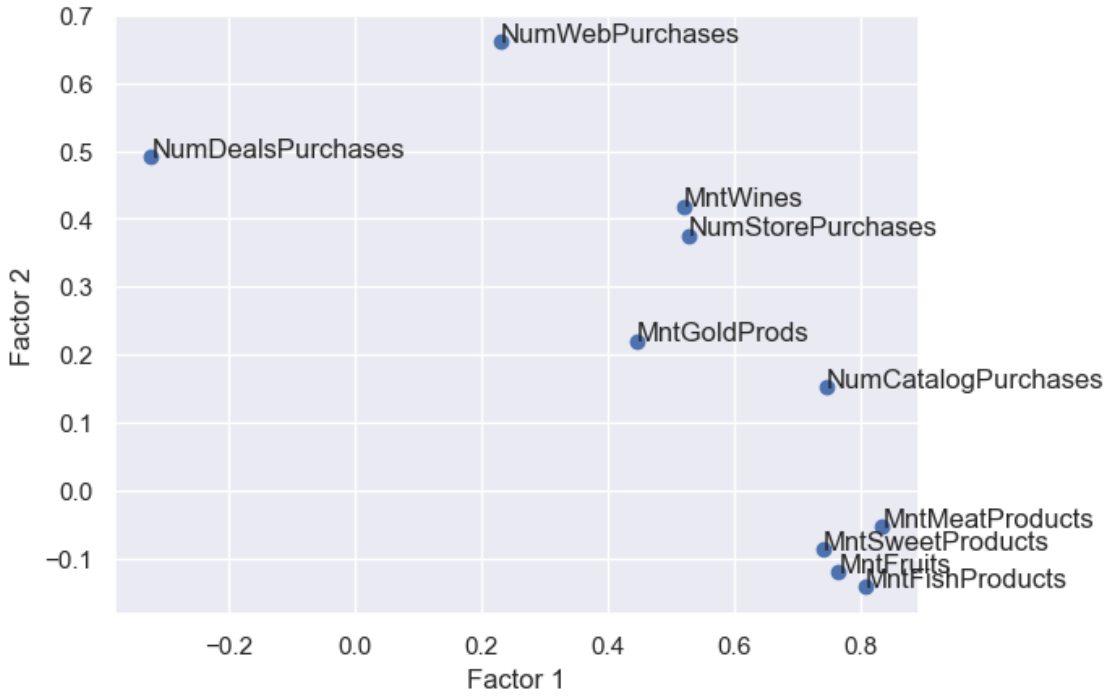
[94]:
```
# Crear un objeto FactorAnalyzer
fa = FactorAnalyzer(n_factors=2, rotation='promax')

# Realizar el análisis factorial en el subset de datos
fa.fit(df_subset)

# Obtener los loadings o cargas factoriales
loadings = fa.loadings_

# Imprimir los loadings para cada variable
print("Cargas factoriales por variable:")
```

```
for i, variable in enumerate(variables):
    print(f"{variable}: {loadings[i]}")
```

```
Cargas factoriales por variable:
MntWines: [0.52020802 0.41845225]
MntFruits: [ 0.7660048  -0.12043695]
MntMeatProducts: [ 0.83492667 -0.05323597]
MntFishProducts: [ 0.80814508 -0.14247865]
MntSweetProducts: [ 0.74155989 -0.08693979]
MntGoldProds: [0.44592067 0.21994104]
NumDealsPurchases: [-0.32189024  0.49196768]
NumWebPurchases: [0.23079347 0.66050552]
NumCatalogPurchases: [0.7471291  0.15173406]
NumStorePurchases: [0.52818445 0.3754792 ]
```

Se puede afirmar, tras observar las cargas factoriales que solo se identifica un factor entre el grupo de variables seleccionado, este agrupa a las variables "MntFruits" (Dinero gastado en frutas), "MntMeatProducts" (Dinero gastado en carne), "MntFishProducts" (Dinero gastado en pescado), "MntSweetProducts" (Dinero gastado en productos dulces), "NumCatalogPurchases" (Numero de compras usando el catalogo). Este factor va a ser llamado "Comportamiento de Compra de Alimentos y Uso del Catálogo". A continuación, se mostrarán gráficamente las cargas factoriales:

[93]:
```
# Crear el gráfico de dispersión de las cargas factoriales
plt.scatter(loadings[:, 0], loadings[:, 1])
plt.xlabel('Factor 1')
plt.ylabel('Factor 2')

# Etiquetar las variables en el gráfico
for i, variable in enumerate(variables):
    plt.annotate(variable, (loadings[i, 0], loadings[i, 1]))

# Mostrar el gráfico
plt.show()
```




Se puede observar en el gráfico de dispersión que en la esquina inferior derecha están agrupadas las 5 variables antes mencionadas, por lo cual en conjunto pueden ser consideradas como un factor 1, pero en el caso del factor 2 que se pensaba que existía, solo se ve una variable (NumWebPurchases) con alta carga en este factor, lo cual no es suficiente para generar un factor dentro del análisis. Finalmente, debido a que "MntWines", "NumStorePurchases" y "MntGoldProds" estan practicamente al centro del grafico (sus cargas estan repartidas de tal forma que no aportan a ningun factor de forma significativa), es posible afirmar que estas variables NO aportan informacion al analisis.

[90]:
```
print(semopy.efa.explore_cfa_model(df_subset, pval=0.05))
```

```
eta1 =~ MntMeatProducts + MntFishProducts + MntFruits + MntSweetProducts +
MntWines + MntGoldProds
```

### 1.4.1 Pregunta 5

## 1.5 General CFA

Basándonos en los resultados de EFA, implementamos CFA.

[106]:
```
mod = """
# measurement model
eta1 =~ MntFruits + MntMeatProducts + MntFishProducts + MntSweetProducts +␣
 ↪NumCatalogPurchases
    """
```

```
model = semopy.Model(mod)
out=model.fit(df)
print(out)
```

```
Name of objective: MLW
Optimization method: SLSQP
Optimization successful.
Optimization terminated successfully
Objective value: 0.181
Number of iterations: 70
Params: 6.229 1.488 1.072 0.080 23703.933 1506.734 863.616 2.991 905.309 747.963
```

[107]:
```
model.inspect(mode='list', what="names", std_est=True)
```

[107]:

| | lval | op | rval | Estimate | Est. Std | Std. Err | z-value | p-value |
|---|---|---|---|---|---|---|---|---|
| 0 | MntFruits | ~ | eta1 | 1.000000 | 0.672617 | - | - | - |
| 1 | MntMeatProducts | ~ | eta1 | 6.228616 | 0.741886 | 0.212697 | 29.284 | 0.0 |
| 2 | MntFishProducts | ~ | eta1 | 1.487541 | 0.723504 | 0.051805 | 28.714486 | 0.0 |
| 3 | MntSweetProducts | ~ | eta1 | 1.071507 | 0.706108 | 0.038053 | 28.157924 | 0.0 |
| 4 | NumCatalogPurchases | ~ | eta1 | 0.079731 | 0.783507 | 0.002616 | 30.472948 | 0.0 |
| 5 | eta1 | ~~ | eta1 | 747.962890 | 1.000000 | 44.887116 | 16.663198 | 0.0 |
| 6 | MntMeatProducts | ~~ | MntMeatProducts | 23703.933450 | 0.449605 | 915.681866 | 25.886647 | 0.0 |
| 7 | MntFishProducts | ~~ | MntFishProducts | 1506.734258 | 0.476542 | 56.526338 | 26.655437 | 0.0 |
| 8 | MntSweetProducts | ~~ | MntSweetProducts | 863.616038 | 0.501411 | 31.644945 | 27.290806 | 0.0 |
| 9 | NumCatalogPurchases | ~~ | NumCatalogPurchases | 2.990633 | 0.386117 | 0.126283 | 23.681982 | 0.0 |
| 10 | MntFruits | ~~ | MntFruits | 905.309049 | 0.547586 | 31.975564 | 28.312528 | 0.0 |

Se puede observar que la variable con mayor importancia en el único factor es “MntFruits”. Tal como se dijo anteriormente, tras el grupo de variables asociado, el factor será llamado: “Comportamiento de Compra de Alimentos y Uso del Catálogo”

[108]:
```
semopy.calc_stats(model)
```

[108]:
```
        DoF  DoF Baseline        chi2  chi2 p-value  chi2 Baseline       CFI  \
Value     5            10  399.214175           0.0    6146.018571  0.935754

             GFI     AGFI       NFI       TLI     RMSEA        AIC        BIC  \
Value  0.935045  0.87009  0.935045  0.871508  0.189136  19.637901  76.622729

         LogLik
Value  0.18105
```

- El valor del CFI es de 0.935, lo que indica un buen ajuste del modelo.
- El valor de TLI es de 0.872 indica un ajuste casi bueno del modelo.

[102]:
```
semopy.semplot(model, "model.png")
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_93140\1726753845.py in <module>
----> 1 semopy.semplot(model, "model.png")

~\AppData\Roaming\Python\Python39\site-packages\semopy\plot.py in semplot(mod,␣
 ↪filename, inspection, plot_covs, plot_exos, images, engine, latshape,␣
 ↪plot_ests, std_ests, show)
     58     """
     59     if not __GRAPHVIZ:
---> 60         raise ModuleNotFoundError("No graphviz module is installed.")
     61     if type(mod) is str:
     62         mod = Model(mod)

ModuleNotFoundError: No graphviz module is installed.
```

Finalmente, dado que solo existe un factor en el análisis, no vale la pena calcular la correlación entre factores entre un factor existente y uno que es ficticio, ya que solo tiene a una variable.

### 1.5.1 Pregunta 6

[109]:
```
cor_act_fisica = df_food.corr().apply(lambda s: s.apply('{0:.3f}'.format))
cor_act_fisica["Response"]
```

[109]:
```
Income                0.175
Kidhome              -0.078
Teenhome             -0.155
Recency              -0.200
MntWines              0.246
MntFruits             0.122
MntMeatProducts       0.249
MntFishProducts       0.107
MntSweetProducts      0.115
```

```
MntGoldProds             0.140
NumDealsPurchases        0.005
NumWebPurchases          0.150
NumCatalogPurchases      0.235
NumStorePurchases        0.034
NumWebVisitsMonth       -0.005
AcceptedCmp3             0.254
AcceptedCmp4             0.180
AcceptedCmp5             0.325
AcceptedCmp1             0.297
AcceptedCmp2             0.169
Complain                -0.000
Z_CostContact              nan
Z_Revenue                  nan
Response                 1.000
Age                     -0.021
Customer_Days            0.197
marital_Divorced         0.055
marital_Married         -0.081
marital_Single           0.114
marital_Together        -0.075
marital_Widow            0.045
education_2n Cycle      -0.035
education_Basic         -0.050
education_Graduation    -0.041
education_Master         0.004
education_PhD            0.090
MntTotal                 0.265
MntRegularProds          0.262
AcceptedCmpOverall       0.427
Name: Response, dtype: object
```

Se considerarán las variables que tengan una correlación con "Response" absolutamente mayor o igual que 0.2, a excepción de "MntTotal", "MntRegularProds" y "AcceptedCmpOverall", debido a que son sumas de variables. Así, se incluyen las variables "Recency", "MntWines", "MntMeatProducts", "NumCatalogPurchases", "AcceptedCmp3", "AcceptedCmp5", "AcceptedCmp1".

[112]:
```
df_subset2 = df_food[["Response", "Recency", "MntWines", "MntMeatProducts",
                      "NumCatalogPurchases", "AcceptedCmp3", "AcceptedCmp5",
                      "AcceptedCmp1"]]
df_subset2
```

[112]:

| | Response | Recency | MntWines | MntMeatProducts | NumCatalogPurchases | \ |
|---|---|---|---|---|---|---|
| 0 | 1 | 58 | 635 | 546 | 10 | |
| 1 | 0 | 38 | 11 | 6 | 1 | |
| 2 | 0 | 26 | 426 | 127 | 2 | |
| 3 | 0 | 26 | 11 | 20 | 0 | |

```
4             0       94       173              118                     3
…           …       …        …                …                       …
2200          0       46       709              182                     3
2201          0       56       406               30                     2
2202          0       91       908              217                     3
2203          0        8       428              214                     5
2204          1       40        84               61                     1

      AcceptedCmp3  AcceptedCmp5  AcceptedCmp1
0                0             0             0
1                0             0             0
2                0             0             0
3                0             0             0
4                0             0             0
…              …             …             …
2200             0             0             0
2201             0             0             1
2202             0             0             0
2203             0             0             0
2204             0             0             0

[2205 rows x 8 columns]
```

[115]:
```
# Creación del objeto FactorAnalyzer con rotación promax
fa2 = FactorAnalyzer(rotation='promax')

# Realizar el análisis factorial en el subset de datos
fa2.fit(df_subset2)
```

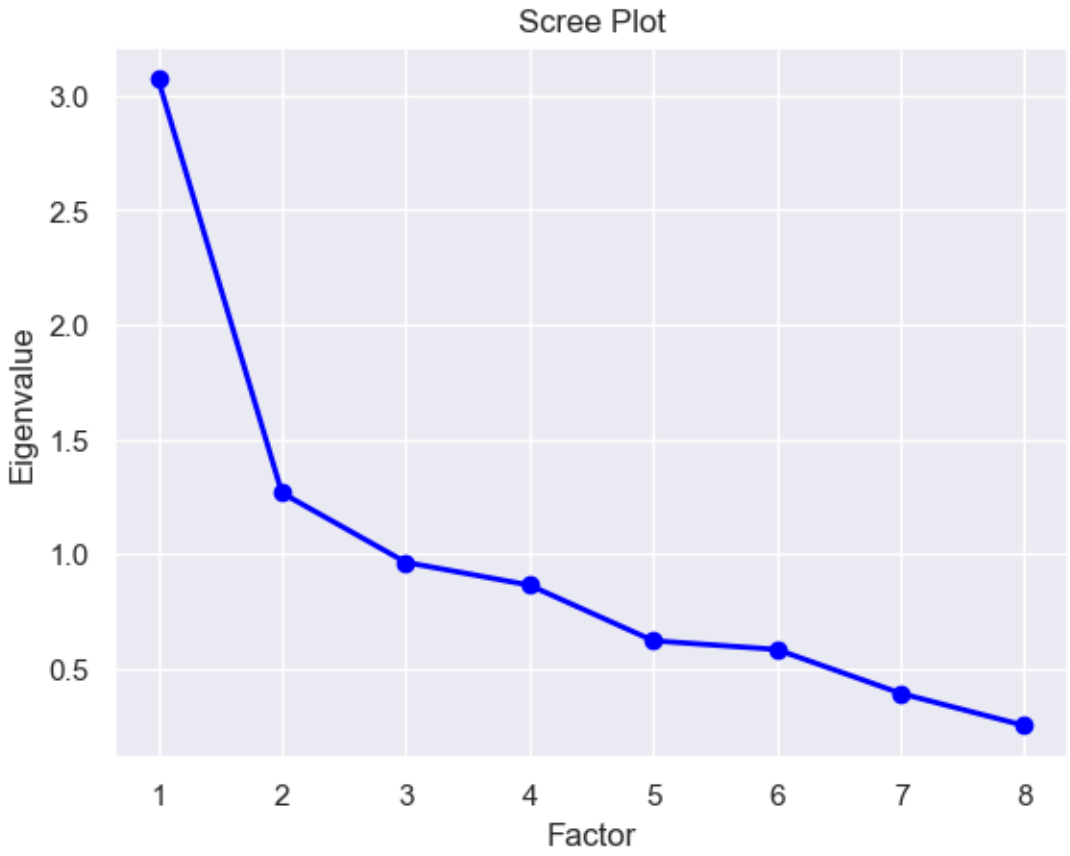
[115]:
```
FactorAnalyzer(rotation_kwargs={})
```

[116]:
```
fa2.get_eigenvalues()
```

[116]:
```
(array([3.07163379, 1.2653054 , 0.96207097, 0.86213322, 0.61947693,
        0.58109177, 0.3896489 , 0.24863902]),
 array([ 2.81647737,  0.94890406,  0.61775519,  0.13461628,  0.01805167,
        -0.02769275, -0.08280445, -0.12121975]))
```

[118]:
```
# Crear un array con los valores del 1 al 8 (correspondiente al número de␣
 ↪factores)
values = np.arange(1, 9)

# Obtener los valores propios del análisis factorial y guardarlos en un␣
 ↪DataFrame
eigenvalues = pd.DataFrame(data=fa2.get_eigenvalues())

# Trazar el gráfico de Scree Plot
```

```
plt.plot(values, eigenvalues.iloc[0], 'o-', linewidth=2, color='blue')
plt.title('Scree Plot')
plt.xlabel('Factor')
plt.ylabel('Eigenvalue')
plt.show()
```




Se puede observar gráficamente que se requieren 2 factores, ya que solo estos tienen valores propios superiores a 1.

[122]:
```
variables2 = ["Response", "Recency", "MntWines", "MntMeatProducts",␣
 ↪"NumCatalogPurchases", "AcceptedCmp3", "AcceptedCmp5",
              "AcceptedCmp1"]
# Crear un objeto FactorAnalyzer
fa2 = FactorAnalyzer(n_factors=2, rotation='promax')

# Realizar el análisis factorial en el subset de datos
fa2.fit(df_subset2)

# Obtener los loadings o cargas factoriales
loadings = fa.loadings_
```

```
# Imprimir los loadings para cada variable
print("Cargas factoriales por variable:")
for i, variable in enumerate(variables2):
    print(f"{variable}: {loadings[i]}")
```

```
Cargas factoriales por variable:
Response: [0.52020802 0.41845225]
Recency: [ 0.7660048  -0.12043695]
MntWines: [ 0.83492667 -0.05323597]
MntMeatProducts: [ 0.80814508 -0.14247865]
NumCatalogPurchases: [ 0.74155989 -0.08693979]
AcceptedCmp3: [0.44592067 0.21994104]
AcceptedCmp5: [-0.32189024  0.49196768]
AcceptedCmp1: [0.23079347 0.66050552]
```

Tras observar las cargas factoriales, solo es posible considerar un factor, este correspondiente a las variables: "Response", "Recency", "MntWines", "MntMeatProducts", "NumCatalogPurchases". Así, el factor se llamará: "Comportamiento de compra en el catálogo relacionado con alimentos". A pesar de esto, solo para capturar más información, se considerará un segundo factor asociado a la variable "AcceptedCmp1", llamado "Concordia del cliente con la primera campaña publicitaria"

[130]:
```
mod = """
# measurement model
eta1 =~ Response + Recency + MntWines + MntMeatProducts + NumCatalogPurchases
eta2 =~ AcceptedCmp1
# regression
Response ~ Recency + MntWines + MntMeatProducts + NumCatalogPurchases
"""
desc = mod
data = df_subset2
model = semopy.Model(desc)
res = model.fit(data)
print(model.inspect())
```

```
WARNING:root:Fisher Information Matrix is not PD.Moore-Penrose inverse will be
used instead of Cholesky decomposition. See 10.1109/TSP.2012.2208105.
```

| | lval | op | rval | Estimate | Std. Err | \ |
|---|---|---|---|---|---|---|
| 0 | Recency | ~ | eta1 | 1.542865 | 1.267092 | |
| 1 | MntWines | ~ | eta1 | 460.537969 | 111.783409 | |
| 2 | MntMeatProducts | ~ | eta1 | 307.785120 | 74.684587 | |
| 3 | NumCatalogPurchases | ~ | eta1 | 4.665006 | 1.114872 | |
| 4 | Response | ~ | eta1 | 1.000000 | - | |
| 5 | AcceptedCmp1 | ~ | eta2 | 1.000000 | - | |
| 6 | Response | ~ | Recency | -0.002605 | 0.000251 | |
| 7 | Response | ~ | MntWines | -0.000178 | 0.000076 | |
| 8 | Response | ~ | MntMeatProducts | -0.000270 | 0.000121 | |
| 9 | Response | ~ | NumCatalogPurchases | -0.115582 | 0.031764 | |

```
10      MntMeatProducts  ~~      MntMeatProducts  23775.201448    1.637373
11              Recency  ~~              Recency    829.744971   24.992371
12            MntWines   ~~            MntWines  56956.010114    0.702015
13  NumCatalogPurchases  ~~  NumCatalogPurchases      1.405419    0.120333
14                 eta2  ~~                 eta2      0.002915    0.000906
15                 eta2  ~~                 eta1      0.055090     0.01302
16                 eta1  ~~                 eta1      0.293234    0.141625
17             Response  ~~             Response      0.072601     0.01757
18         AcceptedCmp1  ~~         AcceptedCmp1      0.057271    0.000906

          z-value   p-value
0        1.217643   0.22336
1        4.119913  0.000038
2        4.121133  0.000038
3        4.184343  0.000029
4               -         -
5               -         -
6      -10.393705       0.0
7       -2.341107  0.019227
8       -2.219903  0.026425
9       -3.638808  0.000274
10    14520.33251       0.0
11       33.19993       0.0
12   81132.203504       0.0
13      11.679378       0.0
14         3.2166  0.001297
15       4.231046  0.000023
16       2.070498  0.038406
17       4.132016  0.000036
18      63.191231       0.0
```

[126]:
```
semopy.semplot(model, "semmodel.png")
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_93140\681625555.py in <module>
----> 1 semopy.semplot(mod, "semmodel.png")

~\AppData\Roaming\Python\Python39\site-packages\semopy\plot.py in semplot(mod,␣
 ↪filename, inspection, plot_covs, plot_exos, images, engine, latshape,␣
 ↪plot_ests, std_ests, show)
     58     """
     59     if not __GRAPHVIZ:
---> 60         raise ModuleNotFoundError("No graphviz module is installed.")
     61     if type(mod) is str:
     62         mod = Model(mod)
```

```
ModuleNotFoundError: No graphviz module is installed.
```

[129]:
```
semopy.calc_stats(model)
```

[129]:

| | DoF | DoF Baseline | chi2 | chi2 p-value | chi2 Baseline | CFI | \ |
|---|---|---|---|---|---|---|---|
| Value | 8 | 15 | 490.68378 | 0.0 | 5106.246335 | 0.905193 | |

| | GFI | AGFI | NFI | TLI | RMSEA | AIC | BIC | \ |
|---|---|---|---|---|---|---|---|---|
| Value | 0.903905 | 0.819822 | 0.903905 | 0.822238 | 0.165455 | 25.554935 | 99.635212 | |

| | LogLik |
|---|---|
| Value | 0.222532 |

- El valor del CFI es de 0.905, lo que indica un buen ajuste del modelo.
- El valor de TLI es de 0.822 indica un ajuste casi bueno del modelo.
- Se puede apreciar que las latentes "Comportamiento de compra en el catálogo relacionado con alimentos" y "Concordia del cliente con la primera campaña publicitaria" tienen valor p inferior a 5% por lo que son significativas para el modelo de regresión que tiene como variable dependiente "Response".
- El comportamiento de compra en el catálogo relacionado con alimentos tiene el coeficiente más alto 0.293234, lo que significa que un cambio en el comportamiento de compra en el catálogo relacionado con alimentos tendrá el mayor impacto en la aceptación de la última campaña publicitaria.