In [43]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

In [104…
```python
data = pd.read_csv("brandrating.csv")
print("=================================================================================")
print("Columnas dataset")
print(data.columns)
print("=================================================================================")
print("Estadisticos numericos dataset")
print(data.describe())
print("=================================================================================")
print("Cantidad de nulos por columna")
print(data.isna().sum())
print("=================================================================================")
print("Tipo de variable de cada columna")
print(data.dtypes)
print("=================================================================================")
print("Tamaño dataset")
print(data.shape)
print("=================================================================================")
print("Dataset")
data.head()
```

```
=================================================================================
Columnas dataset
Index(['perform', 'leader', 'latest', 'fun', 'serious', 'bargain', 'value',
       'trendy', 'rebuy', 'brand'],
      dtype='object')
=================================================================================
Estadisticos numericos dataset
           perform       leader       latest         fun      serious  \
count  1000.000000  1000.000000  1000.000000  1000.00000  1000.000000
mean      4.488000     4.417000     6.195000     6.06800     4.323000
std       3.203454     2.608432     3.078059     2.74425     2.778199
min       1.000000     1.000000     1.000000     1.00000     1.000000
25%       1.000000     2.000000     4.000000     4.00000     2.000000
50%       4.000000     4.000000     7.000000     6.00000     4.000000
75%       7.000000     6.000000     9.000000     8.00000     6.000000
max      10.000000    10.000000    10.000000    10.00000    10.000000

           bargain       value       trendy        rebuy
count  1000.000000  1000.00000  1000.000000  1000.000000
mean      4.259000     4.33700     5.220000     3.727000
std       2.667027     2.39858     2.742101     2.544592
min       1.000000     1.00000     1.000000     1.000000
25%       2.000000     2.00000     3.000000     1.000000
50%       4.000000     4.00000     5.000000     3.000000
75%       6.000000     6.00000     7.000000     5.000000
max      10.000000    10.00000    10.000000    10.000000
=================================================================================
Cantidad de nulos por columna
perform    0
leader     0
latest     0
fun        0
serious    0
bargain    0
value      0
trendy     0
rebuy      0
brand      0
dtype: int64
=================================================================================
Tipo de variable de cada columna
perform     int64
leader      int64
latest      int64
fun         int64
serious     int64
bargain     int64
value       int64
trendy      int64
rebuy       int64
brand      object
dtype: object
=================================================================================
Tamaño dataset
(1000, 10)
=================================================================================
Dataset
```

Out[104…

	perform	leader	latest	fun	serious	bargain	value	trendy	rebuy	brand
0	2	4	8	8	2	9	7	4	6	a
1	1	1	4	7	1	1	1	2	2	a
2	2	3	5	9	2	9	5	1	6	a
3	1	6	10	8	3	4	5	2	1	a
4	1	1	5	8	1	9	9	1	1	a

In [105…
```python
data["brand"].value_counts()
```

Out[105…
```
c    100
h    100
j    100
d    100
g    100
a    100
e    100
b    100
f    100
i    100
Name: brand, dtype: int64
```

In [106…
```python
# Se reemplazan las marcas por un identificador numerico para poder trabajarlas de forma más comoda
dic = {
    "a": 1,
    "b": 2,
    "c": 3,
    "d": 4,
    "e": 5,
    "f": 6,
    "g": 7,
    "h": 8,
    "i": 9,
    "j": 10,
}

data2 = data
data2["brand_num"] = data2["brand"].apply(lambda x: dic[x])
del data2["brand"]
data2.head()
```

Out[106…

	perform	leader	latest	fun	serious	bargain	value	trendy	rebuy	brand_num
0	2	4	8	8	2	9	7	4	6	1
1	1	1	4	7	1	1	1	2	2	1
2	2	3	5	9	2	9	5	1	6	1
3	1	6	10	8	3	4	5	2	1	1
4	1	1	5	8	1	9	9	1	1	1

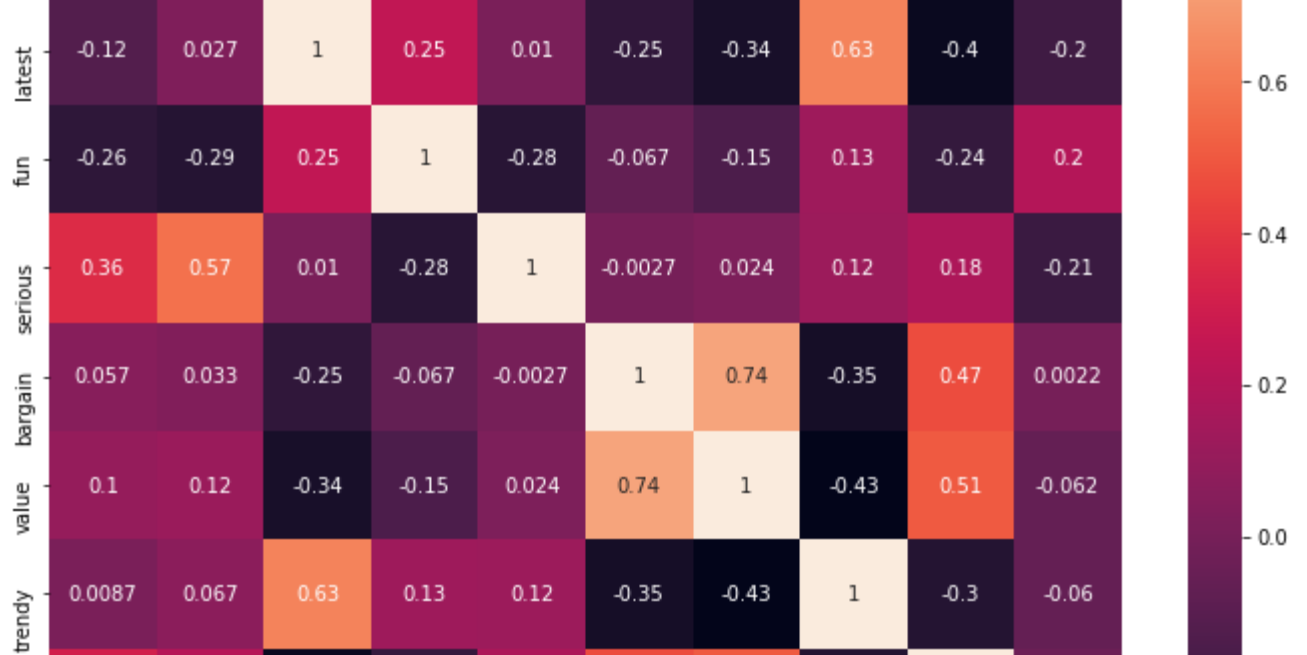
In [107…
```python
data.head()
```

Out[107…

	perform	leader	latest	fun	serious	bargain	value	trendy	rebuy	brand_num
0	2	4	8	8	2	9	7	4	6	1
1	1	1	4	7	1	1	1	2	2	1
2	2	3	5	9	2	9	5	1	6	1
3	1	6	10	8	3	4	5	2	1	1
4	1	1	5	8	1	9	9	1	1	1

Mapa de calor por correlacion de variables del dataset

In [108…
```python
Var_Corr = data2.corr()
plt.figure(figsize = (12,10))
sns.heatmap(Var_Corr, xticklabels=Var_Corr.columns, yticklabels=Var_Corr.columns, annot=True)
plt.show()
```



Componentes principales

In [109…
```python
from sklearn.preprocessing import StandardScaler #estandarizar las variables
from numpy.linalg import eig #eginvalues
import numpy.linalg as la

# Se copian las columnas numericas a un nuevo dataset
data_numerico = data2
scaler = StandardScaler()
data_standarizada = scaler.fit_transform(data_numerico)# Estandarizacion de variables
print(data_standarizada)
```

```
[[-0.77705029 -0.15994618  0.58670187 ... -0.44513691  0.89371412
  -1.5666989 ]
 [-1.08936954 -1.31063813 -0.71346848 ... -1.17486954 -0.679034
  -1.5666989 ]
 [-0.77705029 -0.54351016 -0.38842589 ... -1.53973586  0.89371412
  -1.5666989 ]
 ...
 [-1.08936954 -1.31063813  1.23678704 ... -0.08027059 -0.679034
   1.5666989 ]
 [-1.08936954 -1.31063813  0.26165928 ... -0.08027059 -1.07222103
   1.5666989 ]
 [ 0.78454595 -0.15994618  0.26165928 ... -0.08027059 -1.07222103
   1.5666989 ]]
```

In [110…
```python
# Se revisa si la estandarizacion esta correcta
promedio = np.round(np.mean(data_standarizada), decimals = 2)
desv_std = np.round(np.std(data_standarizada), decimals = 2)
print("Promedio: ", promedio)
print("Desviacion estandar:", desv_std)
```

```
Promedio:  0.0
Desviacion estandar: 1.0
```

In [112…
```python
# Se transforma a un dataframe los campos estandarizados
data_std = pd.DataFrame(data_standarizada, columns = ["perform","leader", "latest", "fun", "serious", "bargain'
data_std.head()
```

Out[112…

	perform	leader	latest	fun	serious	bargain	value	trendy	rebuy	brand_num
0	-0.77705	-0.159946	0.586702	0.704370	-0.836572	1.778524	1.110796	-0.445137	0.893714	-1.566699
1	-1.08937	-1.310638	-0.713468	0.339789	-1.196697	-1.222571	-1.391936	-1.174870	-0.679034	-1.566699
2	-0.77705	-0.543510	-0.388426	1.068950	-0.836572	1.778524	0.276552	-1.539736	0.893714	-1.566699
3	-1.08937	0.607182	1.236787	0.704370	-0.476446	-0.097160	0.276552	-1.174870	-1.072221	-1.566699
4	-1.08937	-1.310638	-0.388426	0.704370	-1.196697	1.778524	1.945040	-1.539736	-1.072221	-1.566699

In [113…
```python
# Se calcula la covarianza
data_cov = np.cov(data_std)
print("=================================================================")
print("Matriz de Covarianza de dataset normalizado")
print(data_cov)
```

```
================================================================
Matriz de Covarianza de dataset normalizado
[[ 1.09466183  0.20194827  1.05947219 ...  0.23680374 -0.44311082
  -0.71461636]
 [ 0.20194827  0.29757813  0.29583521 ...  0.18553548 -0.03040368
   0.05079778]
 [ 1.05947219  0.29583521  1.27657792 ...  0.22718533 -0.51012673
  -0.61752595]
 ...
 [ 0.23680374  0.18553548  0.22718533 ...  1.25340345  0.74105631
   0.33039583]
 [-0.44311082 -0.03040368 -0.51012673 ...  0.74105631  0.84209861
   0.57487248]
 [-0.71461636  0.05079778 -0.61752595 ...  0.33039583  0.57487248
   0.80725313]]
```

In [114…
```python
# Se obtienen los eigenvalues y eigenvectors
eigvalues, eigvectors = eig(data_cov)
print("=================================================================")
print("Primeros 10 Eiginvalues")
print(eigvalues[:10])
print("=================================================================")
print("Cantidad total de Eiginvectors generados")
print(eigvectors.shape)
```

```
=================================================================
Primeros 10 Eiginvalues
[3.16522967e+02+0.00000000e+00j 2.30598439e+02+0.00000000e+00j
 1.26317091e+02+0.00000000e+00j 7.37753697e+01+0.00000000e+00j
 6.58211457e+01+0.00000000e+00j 5.83935180e+01+0.00000000e+00j
 3.94492224e+01+0.00000000e+00j 3.38736904e+01+0.00000000e+00j
 2.73712792e+01+0.00000000e+00j 2.05532926e-14+1.07254672e-14j]
=================================================================
Cantidad total de Eiginvectors generados
(1000, 1000)
```

In [115…
```python
# Calculo de componentes principales
from sklearn.decomposition import PCA
pca = PCA(n_components=9) #cuantas dimensiones de PCA
pca.fit(data_standarizada)
```

Out[115…
```
PCA(n_components=9)
```

In [116…
```python
# Se convierten los datos a las nuevas dimensiones
pca_n = pca.transform(data_standarizada)
print(pca_n)
print(pca_n.shape)
```

```
[[ 1.06413413 -1.03504335  2.64061296 ...  0.34714137  0.02526053
  -0.17680799]
 [-1.49834828 -1.27654725 -0.34434741 ...  0.71645067 -0.49746236
  -0.0131829 ]
 [ 1.24347711 -1.74962785  1.73710924 ...  0.65690248 -0.3951427
  -0.28229882]
 ...
 [-1.54575403 -2.36388306  0.78784886 ... -0.58325887  0.11448226
  -0.7132132 ]
 [-2.19258189 -1.53718259 -1.33955428 ... -0.37630391  0.37614216
  -0.49617375]
 [-1.58343345 -0.03061885 -1.71535243 ... -0.65232979 -0.11110857
  -0.38421631]]
(1000, 9)
```

In [117…
```python
print('Varianza explicada PCA: ', pca_n.shape)
expl = pca.explained_variance_ratio_
print("=================================================================")
print("% de datos faltantes por cada componente: ")
for i, elem in enumerate(expl):
    print((np.round(elem, decimals = 4) * 100), "% faltante para la componente principal n°", i)
print("=================================================================")
print("Porción de la varianza explicada acumulada hasta la 3° componente: ", (np.round(sum(expl[0:3]), decimals
```

```
Varianza explicada PCA:  (1000, 9)
================================================================
% de datos faltantes por cada componente:
29.84 % faltante para la componente principal n° 0
22.14 % faltante para la componente principal n° 1
11.52 % faltante para la componente principal n° 2
9.81 % faltante para la componente principal n° 3
6.64 % faltante para la componente principal n° 4
5.83 % faltante para la componente principal n° 5
5.25 % faltante para la componente principal n° 6
3.51 % faltante para la componente principal n° 7
3.02 % faltante para la componente principal n° 8
================================================================
Porción de la varianza explicada acumulada hasta la 3° componente:  0.6351
```

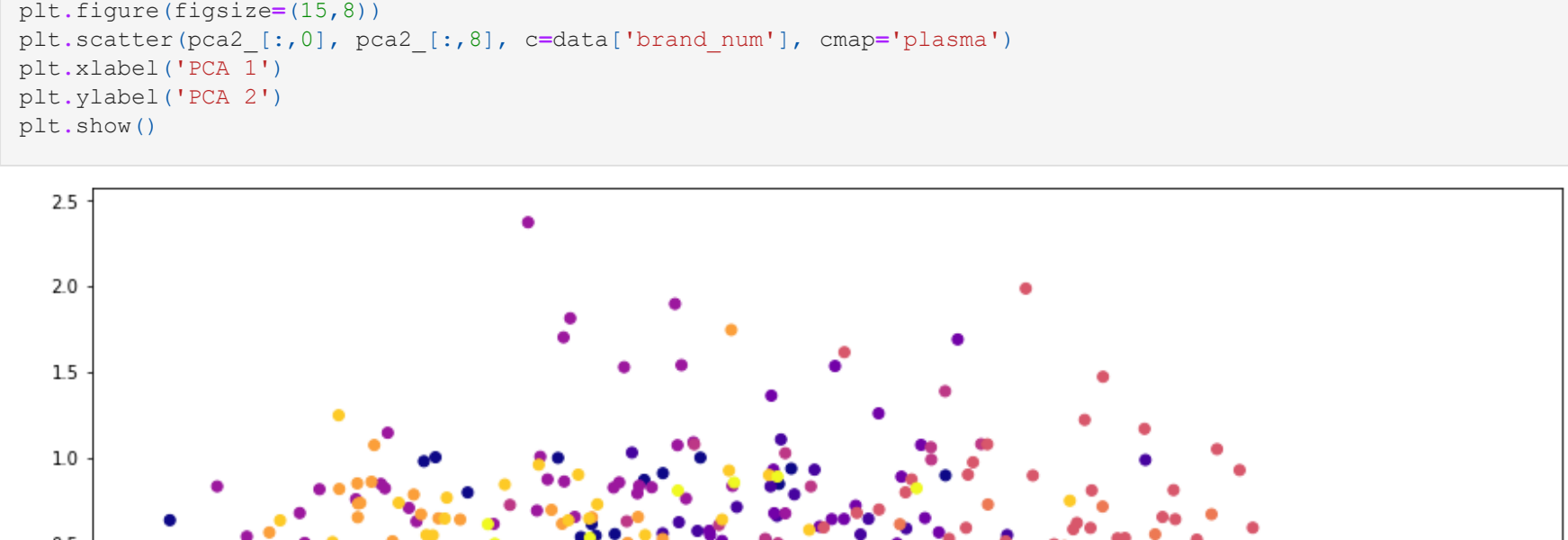
Gráfico de la varianza explicada

In [118…
```python
# graficar la curva de varianza acumulada
plt.figure(figsize=(15,8))
plt.grid()
plt.title("Varianza explicada", fontsize = 20)
plt.xlabel("N° de componentes", fontsize=13)
plt.ylabel("varianza explicada acumulada", fontsize=13)

plt.yticks(fontsize = 13)
plt.xticks(fontsize = 13)

plt.plot(np.cumsum(pca.explained_variance_ratio_), marker="*", linewidth=4.0, alpha = 0.6, c = "#FFC107")

for index,elem in enumerate(np.cumsum(pca.explained_variance_ratio_)):
    value = np.round(elem, decimals = 4)
    plt.text(x = index, y = elem, s = f"{value}")

plt.show()
```

In [119…
```python
# Grafico de PCA
pca2 = PCA(n_components=9)
pca2.fit(data_standarizada)
pca2_ = pca2.transform(data_standarizada)
print("Varianza explicada PCA", pca2_.shape)
expl2 = pca2.explained_variance_ratio_
print(expl2)
print("suma", sum(expl2[0:9]))
```

```
Varianza explicada PCA (1000, 9)
[0.29844063 0.22142569 0.11522607 0.09810949 0.06639711 0.05833695
 0.05253191 0.03505642 0.03023916]
suma 0.9757634296079667
```

In [125…
```python
plt.figure(figsize=(15,8))
plt.scatter(pca2_[:,0], pca2_[:,8], c=data['brand_num'], cmap='plasma')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.show()
```

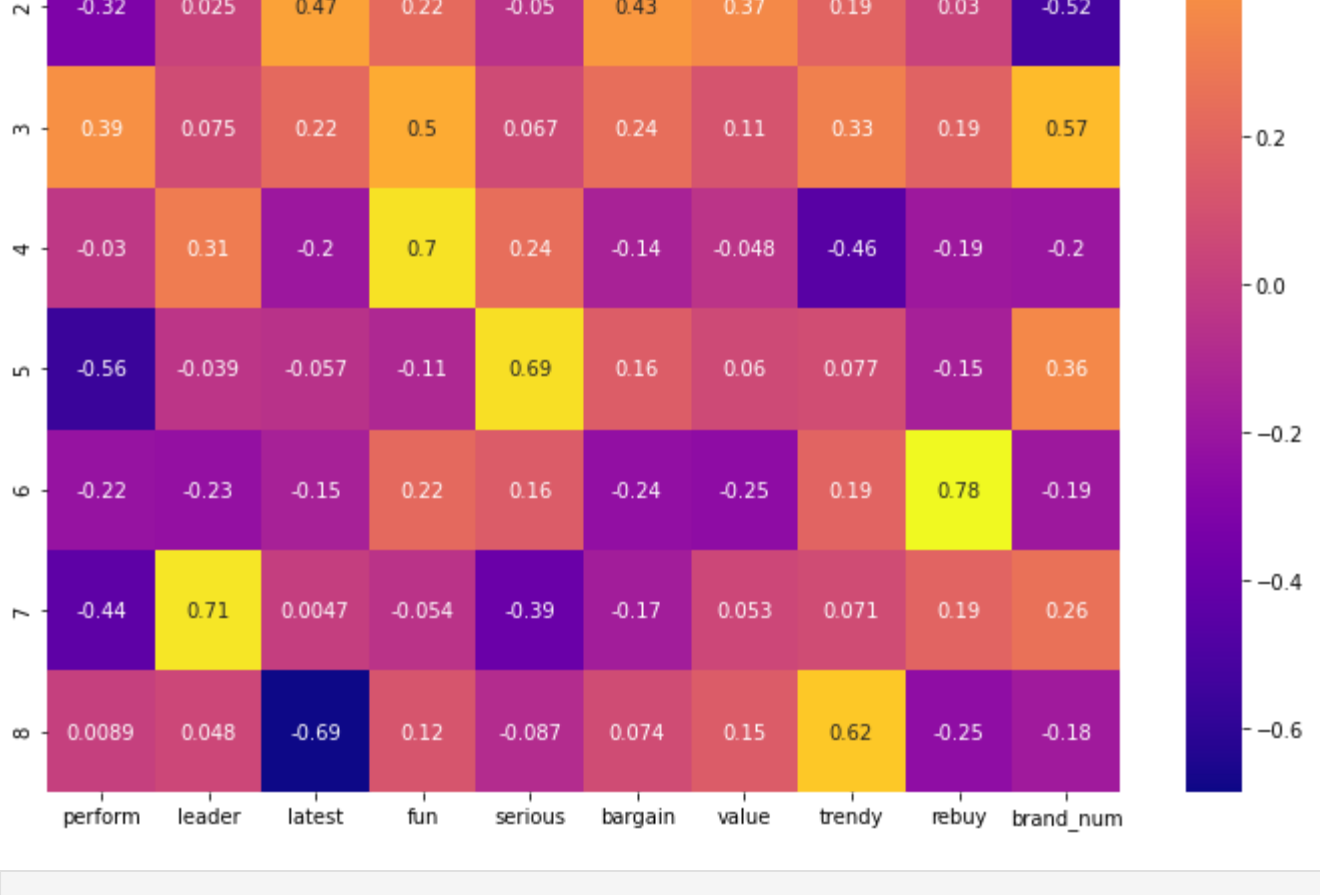


In [126…
```python
data_comp = pd.DataFrame(pca2.components_, columns=["perform","leader", "latest", "fun", "serious", "bargain",
data_comp.head()
```

Out[126…

	perform	leader	latest	fun	serious	bargain	value	trendy	rebuy	brand_num
0	0.242276	0.219316	-0.361506	-0.258003	0.171207	0.394985	0.444711	-0.343514	0.438194	-0.053310
1	0.355107	0.508275	0.238534	-0.242867	0.483063	-0.209249	-0.177254	0.316423	-0.005807	-0.301599
2	-0.315057	0.024712	0.468948	0.219985	-0.050400	0.426112	0.366046	0.193287	0.030007	-0.524844
3	0.394008	0.074565	0.216965	0.503346	0.066820	0.237947	0.113891	0.325426	0.188307	0.568639
4	-0.030330	0.310094	-0.199088	0.704177	0.241633	-0.141564	-0.048185	-0.458381	-0.189803	-0.201330

In [132…
```python
plt.figure(figsize=(12,10))
sns.heatmap(data_comp, cmap='plasma', annot = True)
plt.show()
```



In [ ]: