Chapter 16 - Exercise 3: Marketing Dataset

Cho dữ liệu marketing_data.csv chứa liệu bán hàng: thị phần (market share), thông tin cửa hàng (store variables), thông tin cạnh tranh (competition variables), và dữ liệu trong hoạt động quảng cáo (advertising activity data).

Definition: Private label products are those manufactured by one company for sale under store own brands name sometime also known as white labels.

GRP: Le GRP is an indicator of the advertising pressure of a given media. It corresponds to the average number of advertising contacts obtained on 100 individuals of the targeted target.

Reach: refers to the total number of different people or households exposed, at least once, to a medium during a given period.

Here is a description of the fields in the data set:

- week: the week number
- Year: the data span approximately 3 years from mi 2010 to mid 2013
- Market.Share: the category market share of the product
- Av.Price.per.kg: average price of 1 kilogram of the product
- Non.Promo.Price.per.kg: Non promotional price of the product
- Promo.Vol.Share: ratio of the promotion to. Normal sales
- Total.Weigh: total weight of the product sold
- Share.of.Ean.Weigh:
- Avg.price.vs.PLB: Ratio of price versus the store private brand in the same category.
- Non.promo.price.vs.PLB: average non promotion price ration to the private label brand
- Promo.vol.sh.index.vs.PLB: ratio promotion volume to the private label brand
- Total.cm.shelf: Total of linear space taken by the product in centimeters
- Shelf.share: share of the total shelf taken by the category
- Top.of.mind: ratio interview that cited the brand top of mind. (this is an answer to the question: can you cite some brands in the category X)
- Spontaneous: ratio of interviewees spontaneously citing the brand
- Aided: ratio of the interviewees that recognized the brand by their logo
- Penetration: ratio of the household that bought at least once the brand in the year.
- Competitor: one competitor market share. This is a competitor brand that is of interest in the analysis.
- GRP.radio: GRP of the radio in a given week.
- Reach.radio: Reach of the radio advertising in a given week.
- GRP.TV: GRP of TV advertising
- Reach.TV: reach of TV advertising
- Reach.cinema: Reach of Cinema advertising
- GRP.outdoor: GRP of outdoor advertising
- GRP.print: GRP of Print advertising

- Share.of.spend: share of the marketing budget in these activities in the given week.

Với khá nhiều thông tin, sẽ rất khó để tìm ra insight từ bộ dữ liệu này. Hãy áp dụng thuật toán PCA để trực quan hóa dữ liệu với 2 hoặc 3 thành phần chính. Tìm insight từ các thành phần chính.

In [1]:
```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import svm
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
import seaborn as sns
```

In [2]:
```
import datetime
x1 = datetime.datetime.now()
print(x1)
```

```
2020-10-14 16:57:21.727580
```

In [3]:
```
data = pd.read_csv("marketing_data.csv")
```

In [4]:
```
data.head()
```

Out[4]:

	week	Year	Market.Share	Av.Price.per.kg	Non-Promo Price.per.kg	Promo.Vol.Share	Total.Weigh	Share.of.E
0	19	2010	38.40	7.61	7.77	26.87	84	
1	20	2010	36.80	7.60	7.80	29.42	84	
2	21	2010	35.21	7.63	7.85	27.27	82	
3	22	2010	35.03	7.22	7.76	52.48	88	
4	23	2010	32.37	7.70	7.78	16.11	82	

5 rows × 26 columns

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156 entries, 0 to 155
Data columns (total 26 columns):
week                         156 non-null int64
Year                         156 non-null int64
Market.Share                 156 non-null float64
Av.Price.per.kg              156 non-null float64
Non-Promo Price.per.kg       156 non-null float64
Promo.Vol.Share              156 non-null float64
Total.Weigh                  156 non-null int64
Share.of.Ean.Weigh           156 non-null float64
Avg price.vs.PLB             156 non-null float64
Non.promo.price.vs.PLB       156 non-null float64
Promo.vol.sh.index.vs.PLB    156 non-null float64
Total.cm.shelf               156 non-null float64
Shelf.share                  156 non-null float64
Top.of.mind                  123 non-null float64
Spontaneous                  123 non-null float64
Aided                        123 non-null float64
Penetration                  123 non-null float64
Competitor                   111 non-null float64
GRP.radio                    14 non-null float64
Reach.radio                  14 non-null float64
GRP.TV                       52 non-null float64
Reach.TV                     52 non-null float64
Reach.cinema                 18 non-null float64
GRP.outdoor                  1 non-null float64
GRP.print                    22 non-null float64
Share.of.spend               116 non-null float64
dtypes: float64(23), int64(3)
memory usage: 31.8 KB
```

In [6]:
```
# Kiểm tra dữ liệu null
pd.isnull(data).sum()
```

Out[6]:
```
week                          0
Year                          0
Market.Share                  0
Av.Price.per.kg               0
Non-Promo Price.per.kg        0
Promo.Vol.Share               0
Total.Weigh                   0
Share.of.Ean.Weigh            0
Avg price.vs.PLB              0
Non.promo.price.vs.PLB        0
Promo.vol.sh.index.vs.PLB     0
Total.cm.shelf                0
Shelf.share                   0
Top.of.mind                  33
Spontaneous                  33
Aided                        33
Penetration                  33
Competitor                   45
GRP.radio                   142
Reach.radio                 142
GRP.TV                      104
Reach.TV                    104
Reach.cinema                138
GRP.outdoor                 155
GRP.print                   134
Share.of.spend               40
dtype: int64
```

In [7]:
```
# Cần bỏ đi các cột thiếu nhiều dữ liệu
# Trên 20% dữ liệu thiếu
datasub = data.iloc[:, 2:13]
# bỏ cột week/Year (là cột sẽ tổng hợp theo Group: Year)
```

In [8]:
```
datasub.head(3)
```

Out[8]:

	Market.Share	Av.Price.per.kg	Non-Promo Price.per.kg	Promo.Vol.Share	Total.Weigh	Share.of.Ean.Weigh	pri
0	38.40	7.61	7.77	26.87	84	19.28	
1	36.80	7.60	7.80	29.42	84	18.90	
2	35.21	7.63	7.85	27.27	82	19.11	

In [9]:
```
# Kiểm tra dữ liệu null
pd.isnull(datasub).sum()
```

Out[9]:
```
Market.Share              0
Av.Price.per.kg           0
Non-Promo Price.per.kg    0
Promo.Vol.Share           0
Total.Weigh               0
Share.of.Ean.Weigh        0
Avg price.vs.PLB          0
Non.promo.price.vs.PLB    0
Promo.vol.sh.index.vs.PLB 0
Total.cm.shelf            0
Shelf.share               0
dtype: int64
```

In [10]:
```
# Không còn dữ liệu null
```

In [11]:
```
datasub.shape
```

Out[11]:
```
(156, 11)
```

In [12]:
```
# Xem xét mối liên quan của các thuộc tính khi áp dụng PCA
datasub.corr()
```

Out[12]:

	Market.Share	Av.Price.per.kg	Non-Promo Price.per.kg	Promo.Vol.Share	Total.Weigh
Market.Share	1.000000	-0.324174	0.047841	0.575326	0.063402
Av.Price.per.kg	-0.324174	1.000000	0.717341	-0.336154	0.005114
Non-Promo Price.per.kg	0.047841	0.717341	1.000000	0.281971	0.005141
Promo.Vol.Share	0.575326	-0.336154	0.281971	1.000000	0.047171
Total.Weigh	0.063402	0.005114	0.005141	0.047171	1.000000
Share.of.Ean.Weigh	0.517212	0.040078	0.324106	0.347524	-0.141587
Avg price.vs.PLB	-0.537902	0.713328	0.508281	-0.243109	0.312546
Non.promo.price.vs.PLB	-0.275228	0.616339	0.630557	-0.022154	0.241235
Promo.vol.sh.index.vs.PLB	0.603217	-0.194633	0.274118	0.729234	-0.202382
Total.cm.shelf	0.511062	0.049864	0.291483	0.284730	-0.205174
Shelf.share	0.531925	-0.098341	0.141191	0.302807	-0.232825

In [13]:
```
plt.figure(figsize=(10,10))
sns.heatmap(datasub.corr())
plt.show()
```

In [14]:
```
# Một số biến trong đó có liên quan đến nhau => có thể áp dụng PCA
```

In [15]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(datasub)

# Apply transform to datasub
X_scaler = scaler.transform(datasub)
```

In [16]:
```
from sklearn.decomposition import PCA
```

In [17]:
```
# Make an instance of the Model
pca = PCA(n_components=datasub.shape[1])
```

In [18]:
```
pca.fit(X_scaler)
```

Out[18]:
```
PCA(copy=True, iterated_power='auto', n_components=11, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

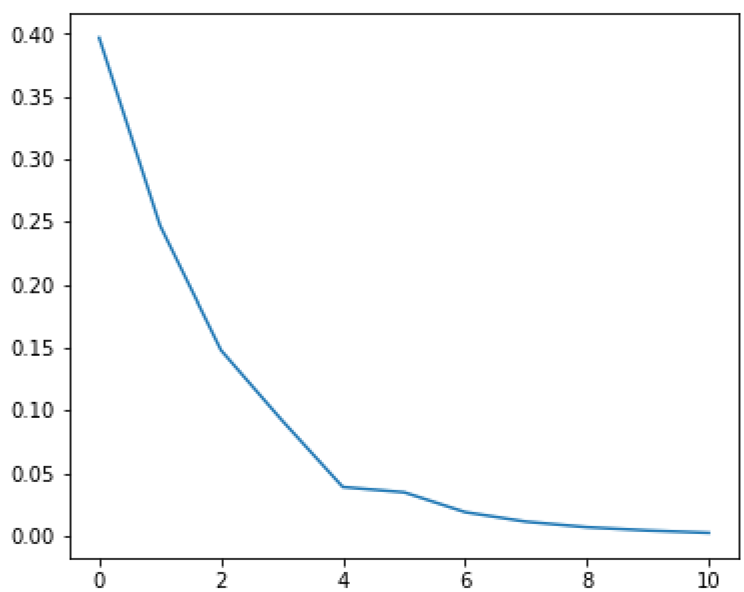
In [19]:
```
pca.n_components_
```

Out[19]:
```
11
```

In [20]:
```
pca.explained_variance_ratio_
```

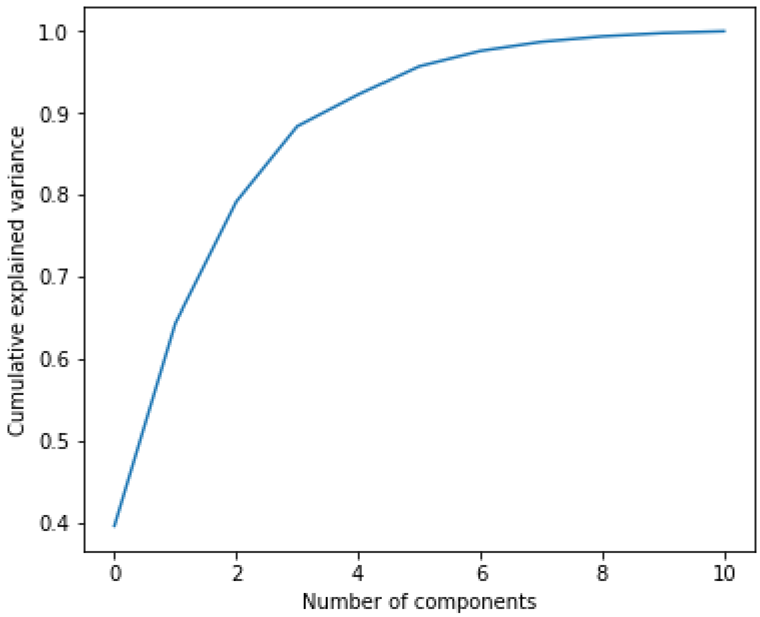
Out[20]:
```
array([0.39684497, 0.24705638, 0.1479238 , 0.0920585 , 0.03877938,
       0.03453391, 0.01872898, 0.0111113 , 0.00669423, 0.00406497,
       0.0022036 ])
```

In [21]:
```
plt.figure(figsize=(6,5))
plt.plot(pca.explained_variance_ratio_)
plt.show()
```



In [22]:
```
plt.figure(figsize=(6,5))
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of components')
plt.ylabel('Cumulative explained variance')
```

Out[22]: `Text(0, 0.5, 'Cumulative explained variance')`



In [23]:
```
# if two components
sum(pca.explained_variance_ratio_[0:2])
```

Out[23]: `0.6439013436160661`

In [24]:
```
# if three components
sum(pca.explained_variance_ratio_[0:3])
```

Out[24]: `0.7918251413353532`

In [25]:
```
# 2 components
pca = PCA(n_components=2)
```

In [26]:
```
principalComponents = pca.fit_transform(X_scaler)
```

In [27]:
```
principalDf = pd.DataFrame(data = principalComponents,
                           columns = ['PC1',
                                      'PC2'])
```

In [28]:
```
principalDf.head(3)
```

Out[28]:

	PC1	PC2
0	3.086575	2.856932
1	2.632491	2.655537
2	2.117565	2.768653

In [29]:
```
principalDf['Year'] = data.Year
```

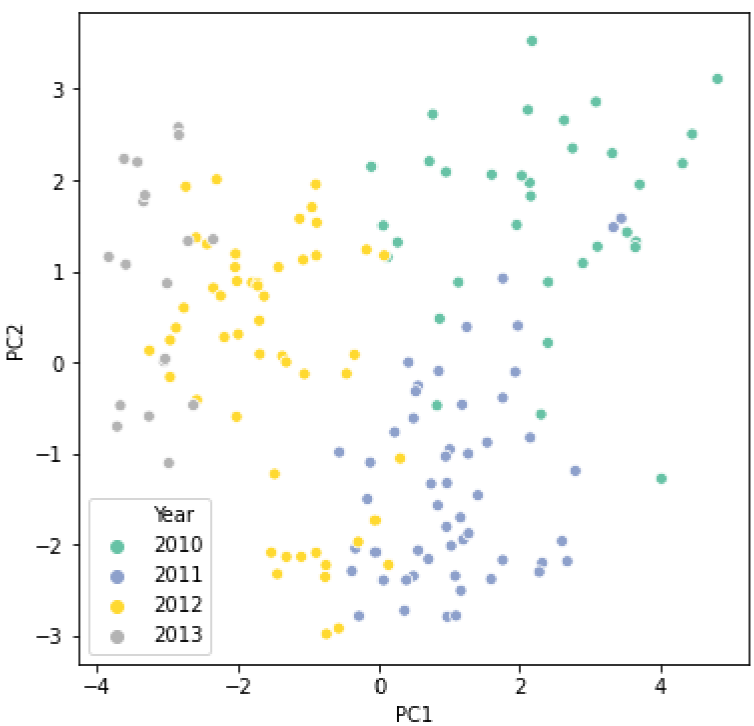
In [30]:
```
principalDf.head(3)
```

Out[30]:

	PC1	PC2	Year
0	3.086575	2.856932	2010
1	2.632491	2.655537	2010
2	2.117565	2.768653	2010

In [31]:
```
import seaborn as sns
```

In [32]:
```
plt.figure(figsize=(6,6))
sns.scatterplot(data=principalDf, x='PC1', y='PC2',
                hue='Year', palette='Set2')
plt.show()
```



In [33]:
```
# 3 components
pca3 = PCA(n_components=3)
```

In [34]:
```
principalComponents3 = pca3.fit_transform(X_scaler)
```

In [35]:
```
principalDf3 = pd.DataFrame(data = principalComponents3,
                           columns = ['PC1',
                                      'PC2',
                                      'PC3'])
```

In [36]:
```
principalDf3.head(3)
```

Out[36]:

	PC1	PC2	PC3
0	3.086575	2.856932	0.947778
1	2.632491	2.655537	1.218877
2	2.117565	2.768653	2.253001

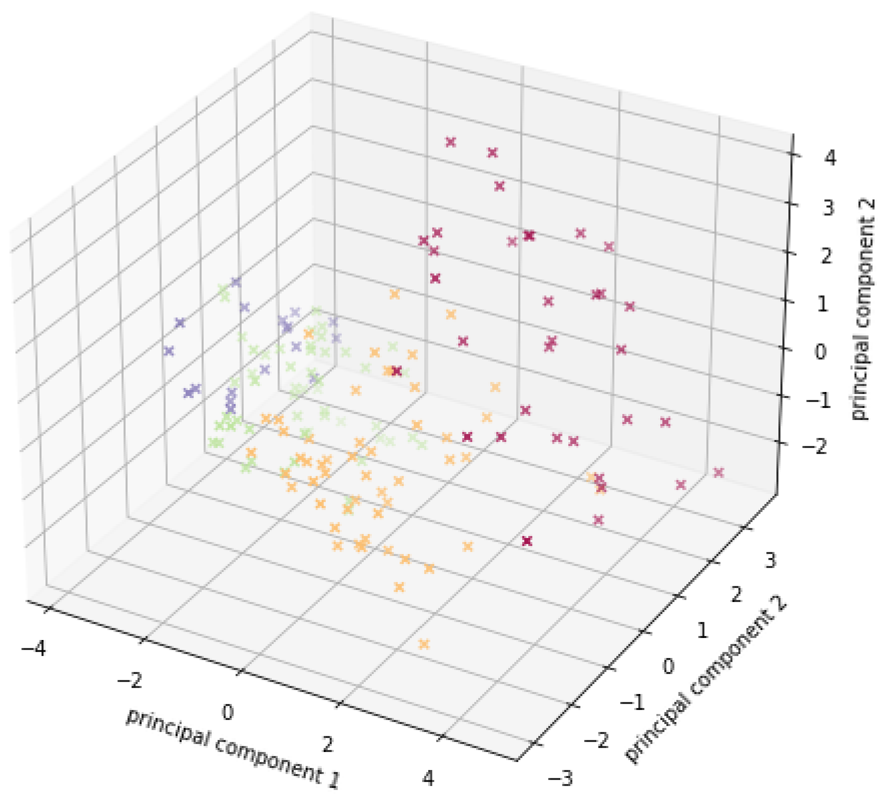
In [37]:
```
principalDf3['Year'] = data.Year
```

In [38]:
```
principalDf3.head(3)
```

Out[38]:

	PC1	PC2	PC3	Year
0	3.086575	2.856932	0.947778	2010
1	2.632491	2.655537	1.218877	2010
2	2.117565	2.768653	2.253001	2010

In [39]:
```
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(principalDf3['PC1'],
           principalDf3['PC2'],
           principalDf3['PC3'],
           c=principalDf3['Year'],
           marker = 'x',
           cmap=plt.cm.Spectral)
ax.set_xlabel('principal component 1')
ax.set_ylabel('principal component 2')
ax.set_zlabel('principal component 2')
plt.show()
```



In [40]:
```
sum(pca3.explained_variance_ratio_)
```

Out[40]:
```
0.7918251413353532
```

Explaining PCA

- The first 3 eigenvectors account for 79.18% of the variance and will be kept.
- Explaining dataset with 3 main components (PCA)

In [41]:
```
principalDf3 = principalDf3.join(datasub)
```

In [42]:
```
principalDf3.head(3)
```

Out[42]:

	PC1	PC2	PC3	Year	Market.Share	Av.Price.per.kg	Non-Promo Price.per.kg	Promo.Vol.Share
0	3.086575	2.856932	0.947778	2010	38.40	7.61	7.77	26.87
1	2.632491	2.655537	1.218877	2010	36.80	7.60	7.80	29.42
2	2.117565	2.768653	2.253001	2010	35.21	7.63	7.85	27.27

In [43]:
```
vects = pca3.components_[:3]
```

Component one

- High: Shelf.share, Total.cm.shelf, Market.Share, Share.of.Ean.Weigh
- Low: Avg price.vs.PLB

In [44]:
```
one = pd.Series(vects[0], index=datasub.columns)
one.sort_values(ascending=False)
```

Out[44]:
```
Shelf.share                   0.406310
Total.cm.shelf                0.374088
Market.Share                  0.369844
Share.of.Ean.Weigh            0.310990
Promo.Vol.Share               0.269878
Promo.vol.sh.index.vs.PLB     0.265543
Non-Promo Price.per.kg       -0.036021
Total.Weigh                  -0.128102
Av.Price.per.kg              -0.227731
Non.promo.price.vs.PLB       -0.297696
Avg price.vs.PLB             -0.399915
dtype: float64
```

Component two

- High: Non-Promo Price.per.kg, Av.Price.per.kg, Non.promo.price.vs.PLB...
- Low: Total.Weigh, Market.Share, Shelf.share

In [45]:
```
two = pd.Series(vects[1], index=datasub.columns)
two.sort_values(ascending=False)
```

Out[45]:
```
Non-Promo Price.per.kg       0.580366
Av.Price.per.kg              0.426798
Non.promo.price.vs.PLB       0.401240
Avg price.vs.PLB             0.294492
Share.of.Ean.Weigh           0.263778
Promo.vol.sh.index.vs.PLB    0.237134
Promo.Vol.Share              0.207890
Total.cm.shelf               0.199172
Shelf.share                  0.110330
Market.Share                 0.106128
Total.Weigh                  0.034143
dtype: float64
```

Component three

- High: Total.cm.shelf, Av.Price.per.kg, Shelf.share...
- Low: Promo.Vol.Share, Promo.vol.sh.index.vs.PLB, Total.Weigh

In [46]:
```
three = pd.Series(vects[2], index=datasub.columns)
three.sort_values(ascending=False)
```

Out[46]:
```
Total.cm.shelf              0.349165
Av.Price.per.kg             0.348326
Shelf.share                 0.316943
Share.of.Ean.Weigh          0.176539
Non-Promo Price.per.kg      0.058696
Avg price.vs.PLB           -0.031666
Non.promo.price.vs.PLB     -0.204910
Market.Share               -0.257276
Total.Weigh                -0.286274
Promo.vol.sh.index.vs.PLB  -0.433281
Promo.Vol.Share            -0.492769
dtype: float64
```