

顧客市場分析X機器學習

Customer Analytics with Machine Learning

這次分析目的

❖主要目的：

1. 利用Python機器學習針對不同特徵進行分類，再利用分類好的群組對加上特定特徵預測消費者購買的品牌。
2. 不同區塊對於價格的反應程度，藉此擬訂行銷策略，提高銷售額以及穩定市場佔有率。

❖此分析有效商家為：

1. **擁有該資料的賣場**：可利用此分析進行促銷、商品陳列位置更改或是對該客群的顧客推薦類似商品以提升營業額。
2. **想要進入該市場的商家**：可藉此資料了解到不同客群所喜歡的品牌樣式，提升商家的品牌定位精準度。

分析項目

1. 依照各區塊族群來進行品牌偏好、銷售額以及購買機率價格彈性分析。

分析資料-1

2000筆不同ID的:

- 性別
- 婚姻
- 年齡
- 教育程度
- 收入
- 職業
- 居住的城市大小

segmentation data.csv(8 features)

ID	Sex	Marital stat	Age	Education	Income	Occupation	Settlement size
100000001	0	0	67	2	124670	1	2
100000002	1	1	22	1	150773	1	2
100000003	0	0	49	1	89210	0	0
100000004	0	0	45	1	171565	1	1
100000005	0	0	53	1	149031	1	1
100000006	0	0	35	1	144848	0	0
100000007	0	0	53	1	156495	1	1
100000008	0	0	35	1	193621	2	1
100000009	0	1	61	2	151591	0	0
100000010	0	1	28	1	174646	2	0
100000011	1	1	25	1	108469	1	0
100000012	1	1	24	1	127596	1	0
100000013	1	1	22	1	108687	1	2
100000014	0	0	60	2	89374	0	0
100000015	1	1	28	1	102899	1	1
100000016	1	1	32	1	88428	0	0
100000017	0	0	53	1	125550	1	0
100000018	0	0	25	0	157434	1	2
100000019	1	1	44	2	261952	2	2
100000020	0	0	31	0	144657	1	1
100000021	0	0	48	1	118777	1	1
100000022	0	0	44	1	147511	1	1
100000023	0	0	48	1	89804	0	0
100000024	0	0	44	1	134918	1	2

分析資料-2

共5萬多筆不同ID的:

- 進入該店的時間
- 有無購買
- 品牌
- 數量
- 上次購買的品牌
- 上次購買品牌的數量
- 品牌價格1~5
- 品牌促銷1~5
- Segment Data的:
 - 性別
 - 婚姻
 - 年齡
 - 教育程度
 - 收入
 - 職業
 - 居住的城市大小

purchase data.csv(24 features)

ID	Day	Incidence	Brand	Quantity	Last_Inc_Bri	Last_Inc_Qu	Price_1	Price_2	Price_3	Price_4	Price_5
200000001	1	0	0	0	0	0	1.59	1.87	2.01	2.09	2.66
200000001	11	0	0	0	0	0	1.51	1.89	1.99	2.09	2.66
200000001	12	0	0	0	0	0	1.51	1.89	1.99	2.09	2.66
200000001	16	0	0	0	0	0	1.52	1.89	1.98	2.09	2.66
200000001	18	0	0	0	0	0	1.52	1.89	1.99	2.09	2.66
200000001	23	0	0	0	0	0	1.5	1.9	1.99	2.09	2.66
200000001	28	1	2	2	0	0	1.5	1.9	1.99	2.09	2.67
200000001	37	0	0	0	2	1	1.5	1.9	1.99	2.09	2.67
200000001	41	0	0	0	0	0	1.35	1.58	1.97	2.09	2.67
200000001	43	0	0	0	0	0	1.35	1.58	1.97	2.09	2.67
200000001	51	0	0	0	0	0	1.35	1.87	1.93	2.09	2.59
200000001	58	1	5	1	0	0	1.39	1.9	1.91	2.12	2.62
200000001	59	0	0	0	5	1	1.39	1.9	1.91	2.12	2.62
200000001	77	0	0	0	0	0	1.36	1.85	1.95	2.12	2.5
200000001	78	0	0	0	0	0	1.47	1.85	1.95	2.12	2.5
200000001	83	0	0	0	0	0	1.47	1.9	1.99	2.12	2.49
200000001	90	0	0	0	0	0	1.39	1.58	1.99	2.12	2.55
200000001	92	0	0	0	0	0	1.39	1.46	1.99	2.12	2.4
200000001	98	0	0	0	0	0	1.47	1.9	1.95	2.12	2.67
200000001	110	1	1	2	0	0	1.47	1.9	1.99	1.97	2.67
200000001	111	0	0	0	1	1	1.47	1.9	1.99	1.97	2.67
200000001	118	0	0	0	0	0	1.43	1.31	1.99	2.16	2.67

Promotion	Promotion	Promotion	Promotion	Promotion	Sex	Marital stati	Age	Education	Income	Occupation	Settlement size
0	1	0	0	0	0	0	47	1	110866	1	0
0	0	0	0	0	0	0	47	1	110866	1	0
0	0	0	0	0	0	0	47	1	110866	1	0
0	0	0	0	0	0	0	47	1	110866	1	0
0	0	0	0	0	0	0	47	1	110866	1	0
0	0	0	0	0	0	0	47	1	110866	1	0
0	0	0	0	0	0	0	47	1	110866	1	0
0	0	0	0	0	0	0	47	1	110866	1	0
1	1	1	0	0	0	0	47	1	110866	1	0
1	1	1	0	0	0	0	47	1	110866	1	0
1	1	1	0	0	0	0	47	1	110866	1	0
1	0	0	0	1	0	0	47	1	110866	1	0
1	0	0	0	0	0	0	47	1	110866	1	0
0	1	0	0	0	0	0	47	1	110866	1	0
0	0	0	0	1	0	0	47	1	110866	1	0
0	0	0	0	0	1	0	47	1	110866	1	0
0	1	0	0	0	0	0	47	1	110866	1	0
0	0	0	0	0	0	0	47	1	110866	1	0
0	0	0	0	0	0	0	47	1	110866	1	0
0	0	0	1	0	0	0	47	1	110866	1	0
1	1	0	0	0	0	0	47	1	110866	1	0
0	1	0	0	0	0	0	47	1	110866	1	0

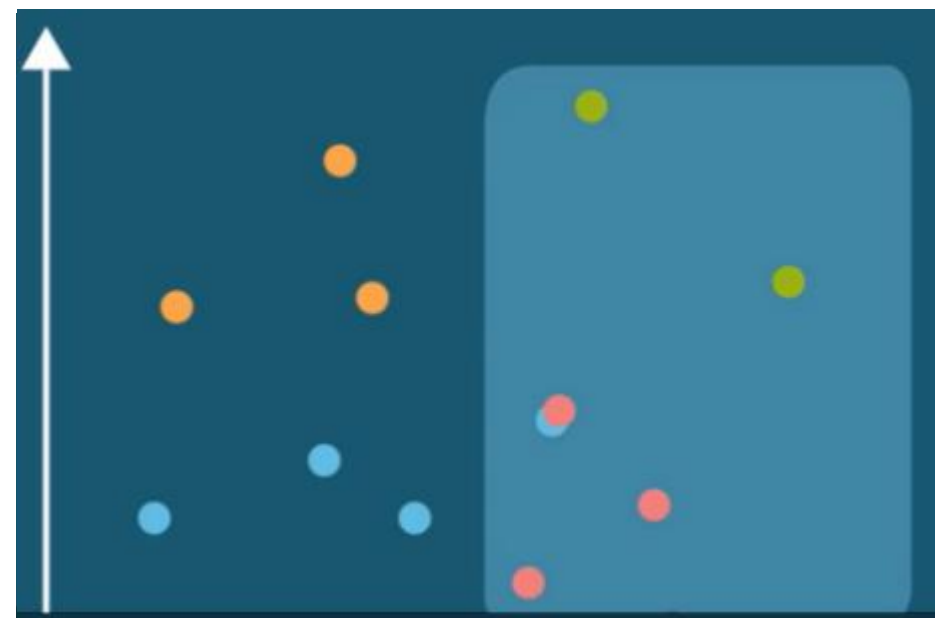
分析工具

- 預處理(Preprocessing): StandardScaler
- 分群(Clustering): linkage、dendrogram(SciPy)、KMeans(scikit-learn)
- 分解(Decomposition): Principal Component Analysis(PCA)
 - 利用PCA轉換維度，將其影響程度略低的component去除。
- 曲線模型(linear_model): LogisticRegression
 - 羅吉斯回歸線為分類演算法，設計來預測分類目標對象(target labels)。
- 製圖: matplotlib、seaborn
- 機器學習演算法: PySpark、Sickit-learn

STP Marketing Model

利用以下方式找出市場利基

- 市場區隔(**S**egmentation): 依照相似的特徵區分為不同族群做區隔
- 市場目標(**T**argeting): 從市場區隔評估潛在利益，決定傳住的區塊
- 市場定位(**P**ositioning): 什麼形式的商品可以吸引到該族群



Hierarchical Clustering

在這個地方立永平衡好的資料針對不同特徵所產生的距離差，利用Ward's linkage method計算出在平衡後各點的距離，進而分類。

- 下一張圖可以看到，下圖紅框的參數得以分出4~6個區塊
- Hierarchical Clustering 比flat clustering (比方說K-means)慢，但是可以先用Hierarchical Clustering做分類(知道有幾個clusters)，再用 flat clustering做 segmentation

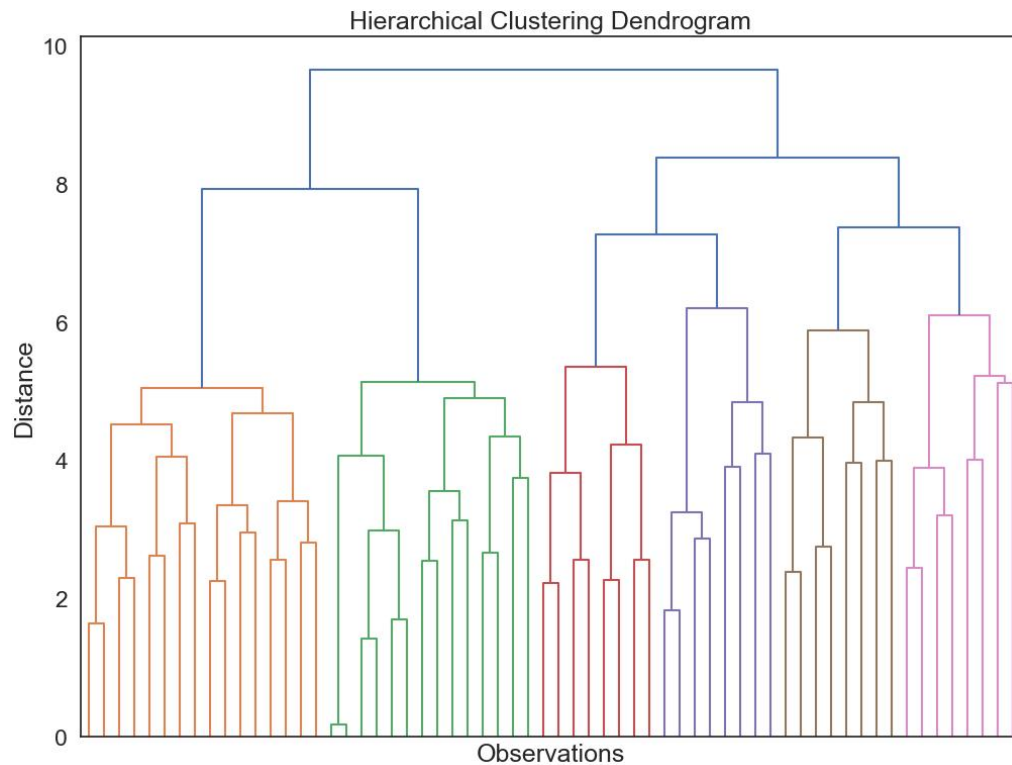
```
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage

scaler = StandardScaler()
segmentation_std = scaler.fit_transform(df_segmentation)
hier_clust = linkage(segmentation_std, method = 'ward')

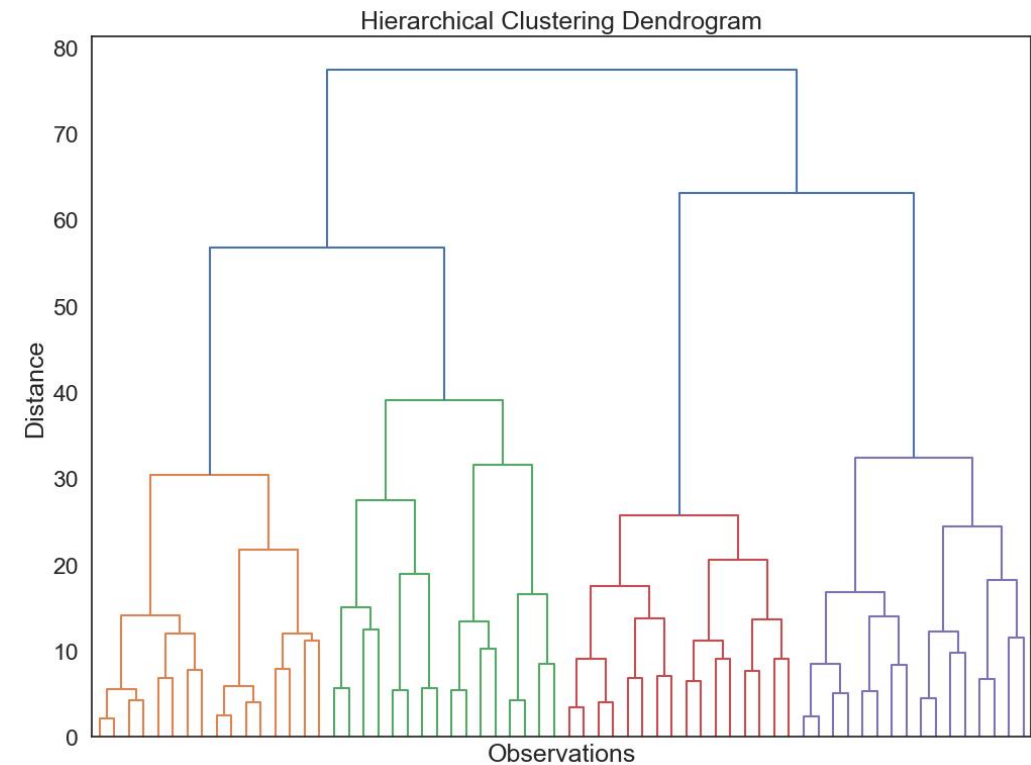
plt.figure(figsize = (12,9))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Observations')
plt.ylabel('Distance')
dendrogram(hier_clust,
            truncate_mode = 'level',
            p = 5,
            show_leaf_counts = False,
            no_labels = True)
plt.show()
```

Standardization & Hierarchical Clustering

method = 'complete'



method = 'ward'



K-means Clustering

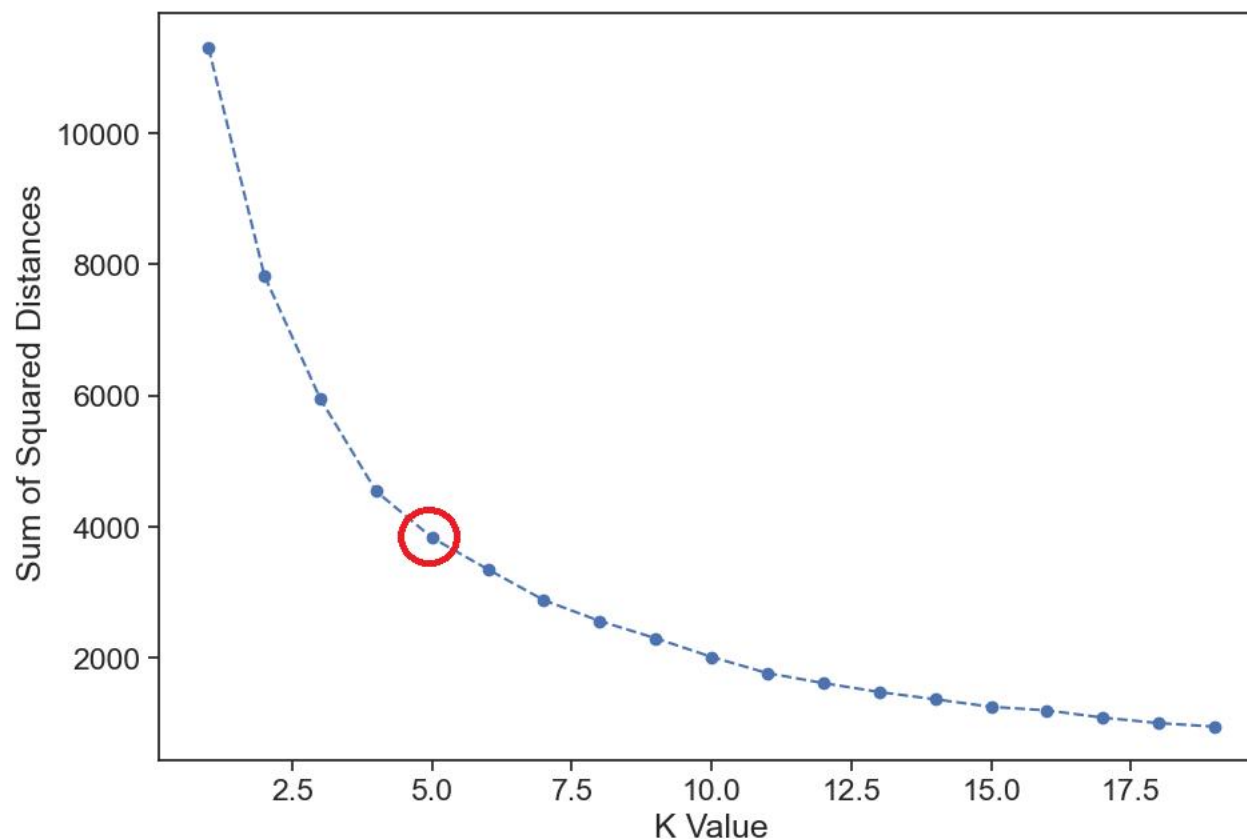
利用上面的演算法已先初步得知群體個數，再來利用K-means做最後的確認。

```
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(segmentation_std)
    wcss.append(kmeans.inertia_)

plt.figure(figsize = (10,8))
plt.plot(range(1, 11), wcss, marker = 'o', linestyle = '--')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.title('K-means Clustering')
plt.show()
```

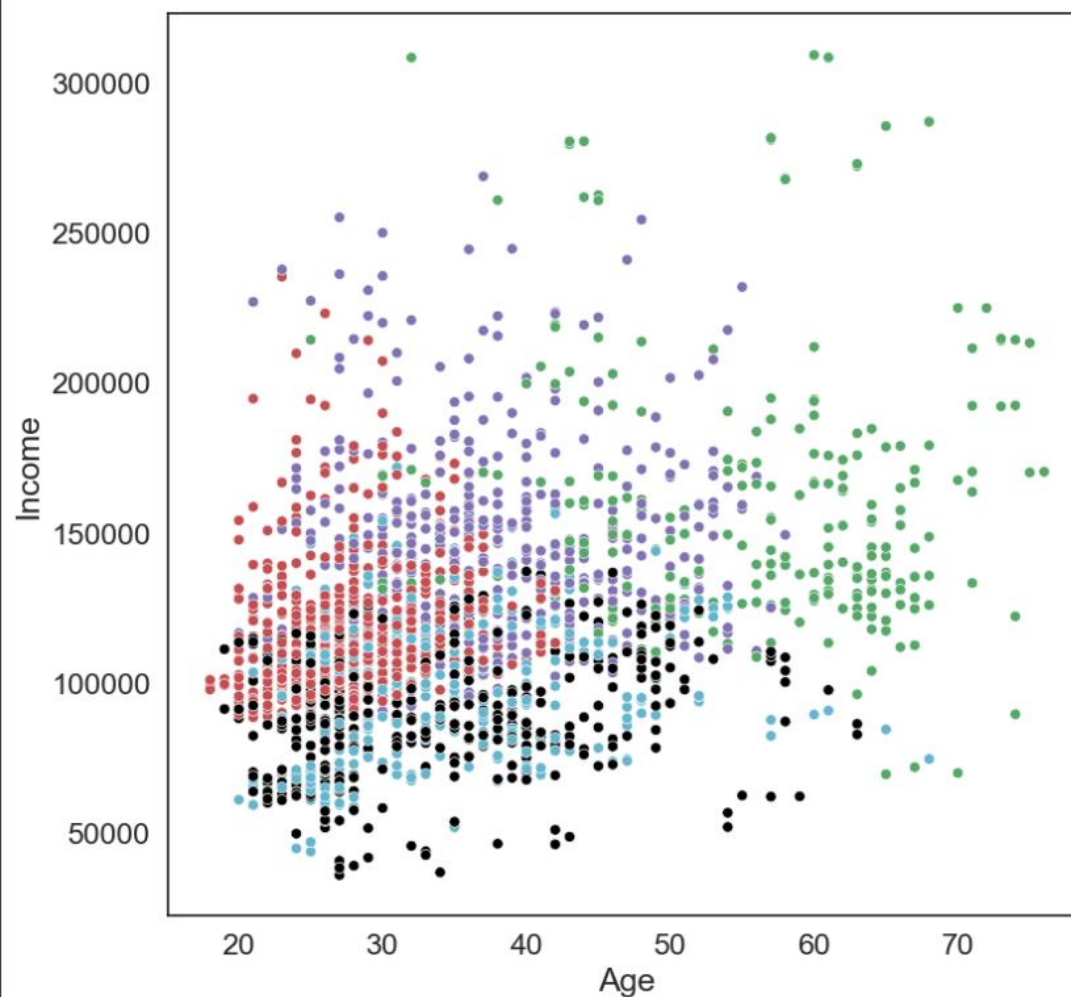
K-means Clustering

利用個點與cluster的平方距離總和 (sum of the squared distances)，在不同總和間找到一個凹折，有一個普遍的方法叫做elbow method，由右圖可以微微的看到紅色圈起來的地方有趨緩的現象，因此可以選擇cluster=5。



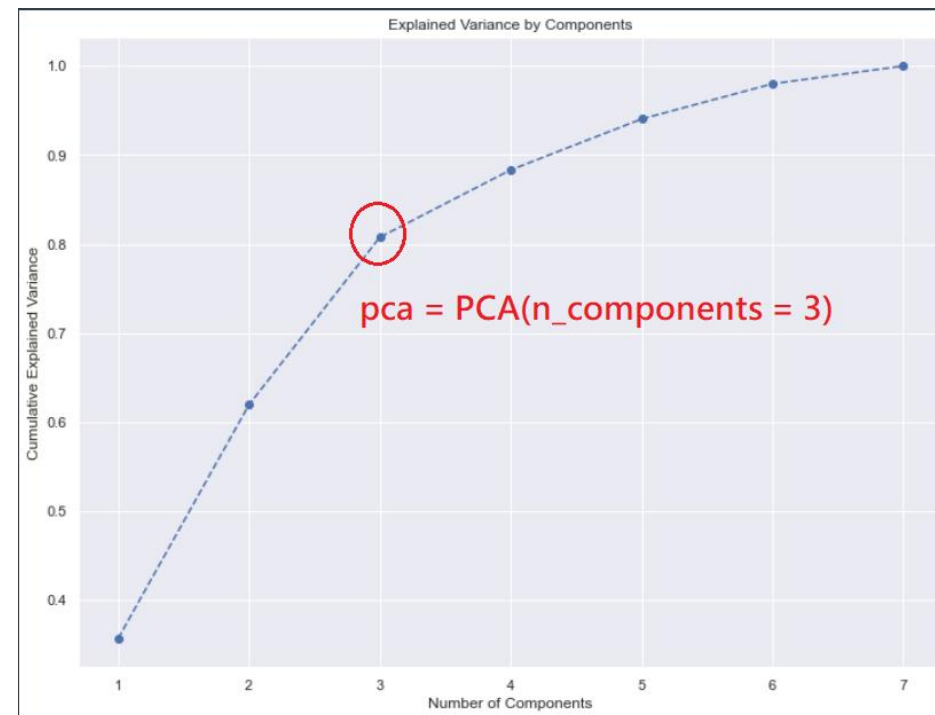
K-means Clustering

最後的表做出來有三個clusters幾乎混在一起，只有綠色的群體有稍微分開，接下來用主成分分析(pca)修正，可以讓分類比較明顯。



Principal Component Analysis(PCA)

- 主成分分析(PCA)以原資料產生出該特徵數量的成分(Component)數，在各個成分數裡，可以選擇變異數(variance)最多的幾個成分，可以做到最佳分類。
- 根據經驗法則，`explained_variance_ratio_`，80%的資料成分數據分析最佳，右圖可以看到利用解釋方差紅色圈起來的地方剛好在0.8左右。



```
#找到PCA components
pca = PCA()
pca.fit(segmentation_std)
print(pca.explained_variance_ratio_)

#圖表顯示，可看出最佳components數
plt.figure(figsize = (12,9))
plt.plot(range(1,8), pca.explained_variance_ratio_.cumsum(), marker = 'o', linestyle = '--')
plt.title('Explained Variance by Components')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
```

PCA Results



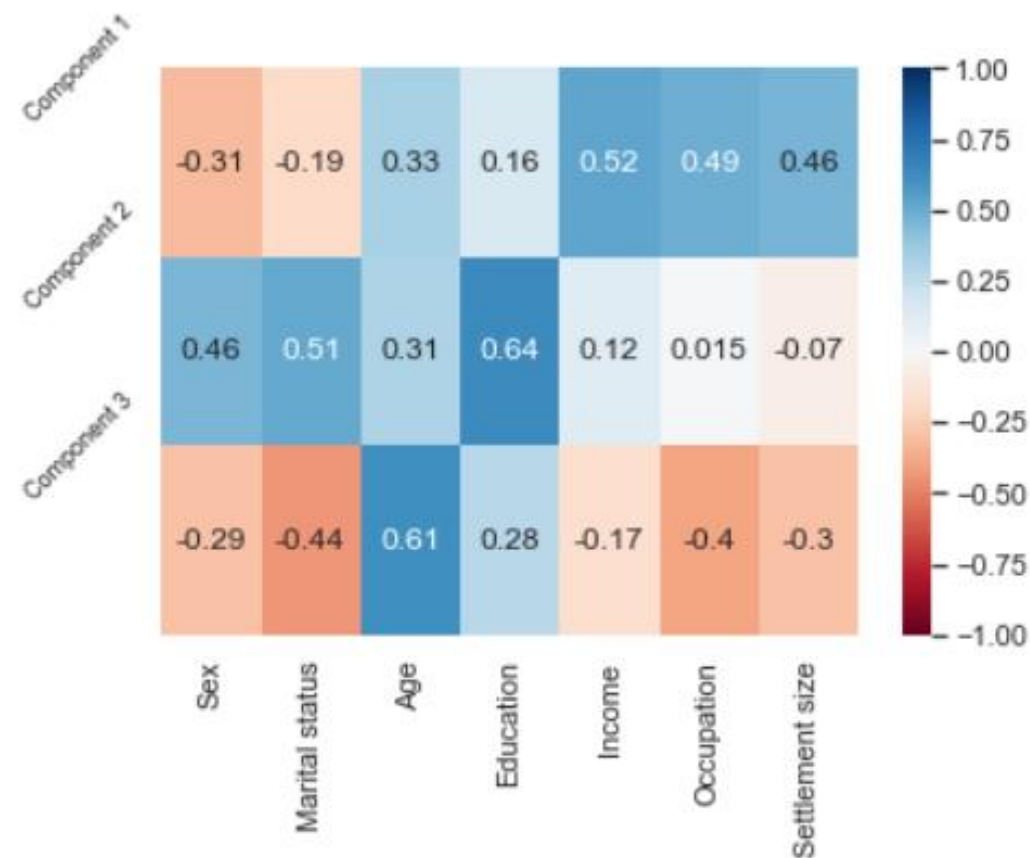
```
#選擇n_components=3
pca = PCA(n_components = 7)
pca.fit(segmentation_std)

#下面了解到components跟原始數據的關係，把它做成heatmap看得更清楚
df_pca_comp = pd.DataFrame(data = pca.components_,
                           columns = df_segmentation.columns.values,
                           index = [f'component {x}' for x in np.arange(1,8)])

plt.figure(figsize=(12,10),dpi=80)
sns.heatmap(df_pca_comp,
            vmin = -1,
            vmax = 1,
            cmap = 'RdBu',
            annot = True)
plt.yticks(np.arange(1,8),
           [f'component {x}' for x in np.arange(1,8)],
           rotation = 45,
           fontsize = 9);
```

依照上一頁的解釋，我們選擇最上面的三個components

PCA Results

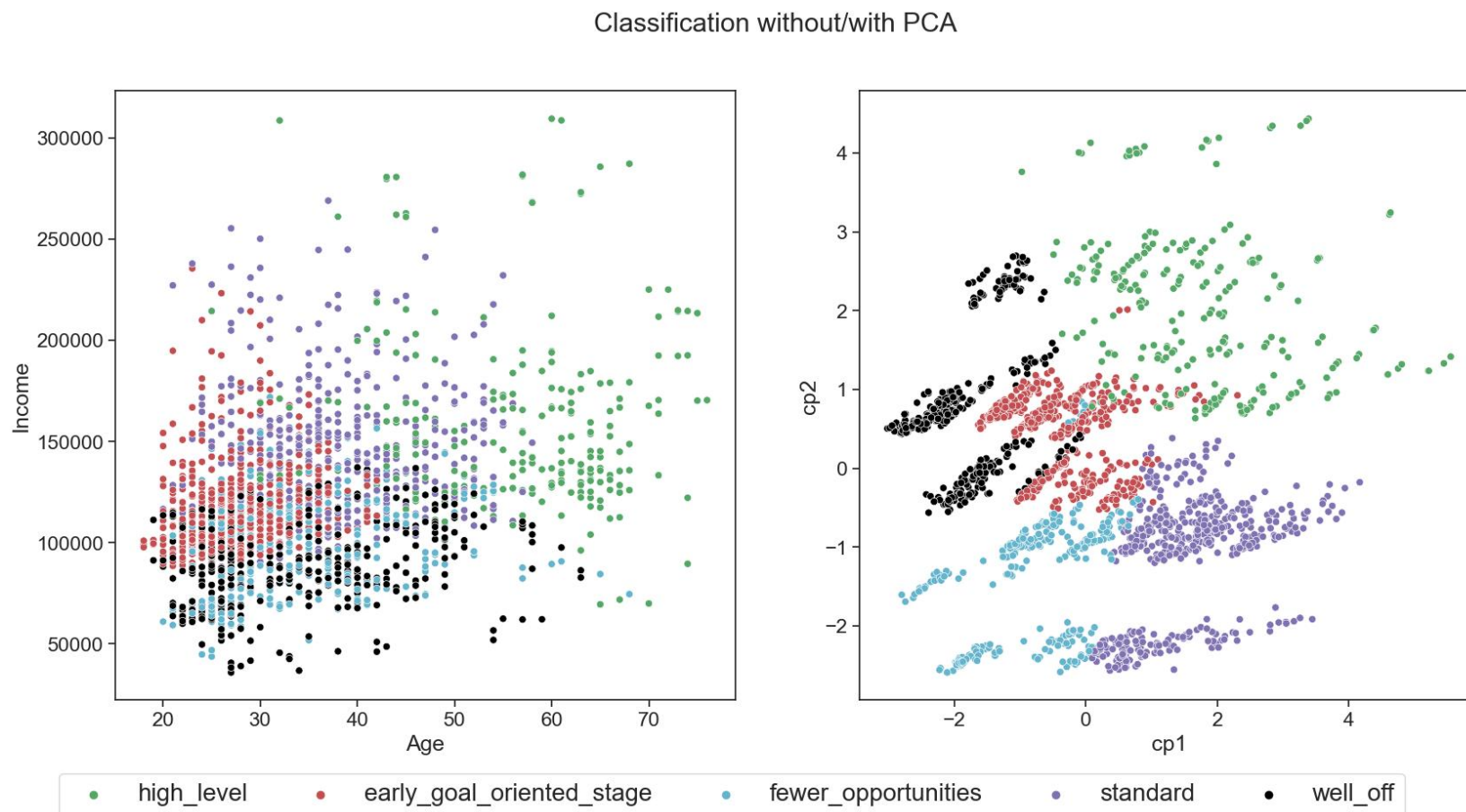


右圖可以看到:

- 越接近0，表示相關越低，這邊看到相關係數在0.5以上較為重要。
- component1 與 Income, ccupation, Settlement size 相關性比較大，這表示它們相互影響力大，所以可推測它們應該是專注在事業上多一點，
- component2 跟 component1 的相關係數相反，所以推測比較專注於學術以及生活方面

K-means clustering with PCA Results

右邊上圖scatterplot可以看到，利用PCA降維後的分群效果比較好。



Pickle

利用pickle儲存我們做好的模型，將在接下來的Purchase Data使用

```
pickle.dump(scaler, open('scaler.pickle', 'wb'))  
pickle.dump(pca, open('pca.pickle', 'wb'))  
pickle.dump(kmeans_pca, open('kmeans_pca.pickle', 'wb'))
```


Purchase Data(分析資料-2)

開始分析各別ID的不同消費行為，共500筆不同ID，

```
df_purchase = pd.read_csv('purchase data.csv')
```

```
df_purchase
```

	ID	Day	Incidence	Brand	Quantity	Last_Inc_Brand	Last_Inc_Quantity	Price_1	Price_2	Price_3	...	Promotion_3	Promotion_4	Promotion_5	Sex	Marital status	Age	Education	Income
0	200000001	1	0	0	0	0	0	1.59	1.87	2.01	...	0	0	0	0	0	47	1	110866
1	200000001	11	0	0	0	0	0	1.51	1.89	1.99	...	0	0	0	0	0	47	1	110866
2	200000001	12	0	0	0	0	0	1.51	1.89	1.99	...	0	0	0	0	0	47	1	110866
3	200000001	16	0	0	0	0	0	1.52	1.89	1.98	...	0	0	0	0	0	47	1	110866
4	200000001	18	0	0	0	0	0	1.52	1.89	1.99	...	0	0	0	0	0	47	1	110866
...
58688	200000500	703	0	0	0	2	1	1.41	1.85	2.01	...	1	0	0	0	0	42	1	120946
58689	200000500	710	0	0	0	0	0	1.36	1.84	2.09	...	0	0	0	0	0	42	1	120946
58690	200000500	717	0	0	0	0	0	1.50	1.80	2.14	...	0	0	0	0	0	42	1	120946
58691	200000500	722	1	2	3	0	0	1.51	1.82	2.09	...	0	0	0	0	0	42	1	120946
58692	200000500	726	0	0	0	2	1	1.51	1.82	2.09	...	0	0	0	0	0	42	1	120946

Purchase Data

事前準備資料，先用pickle
將模型載卸下來，再將該
資料的特徵進行處理

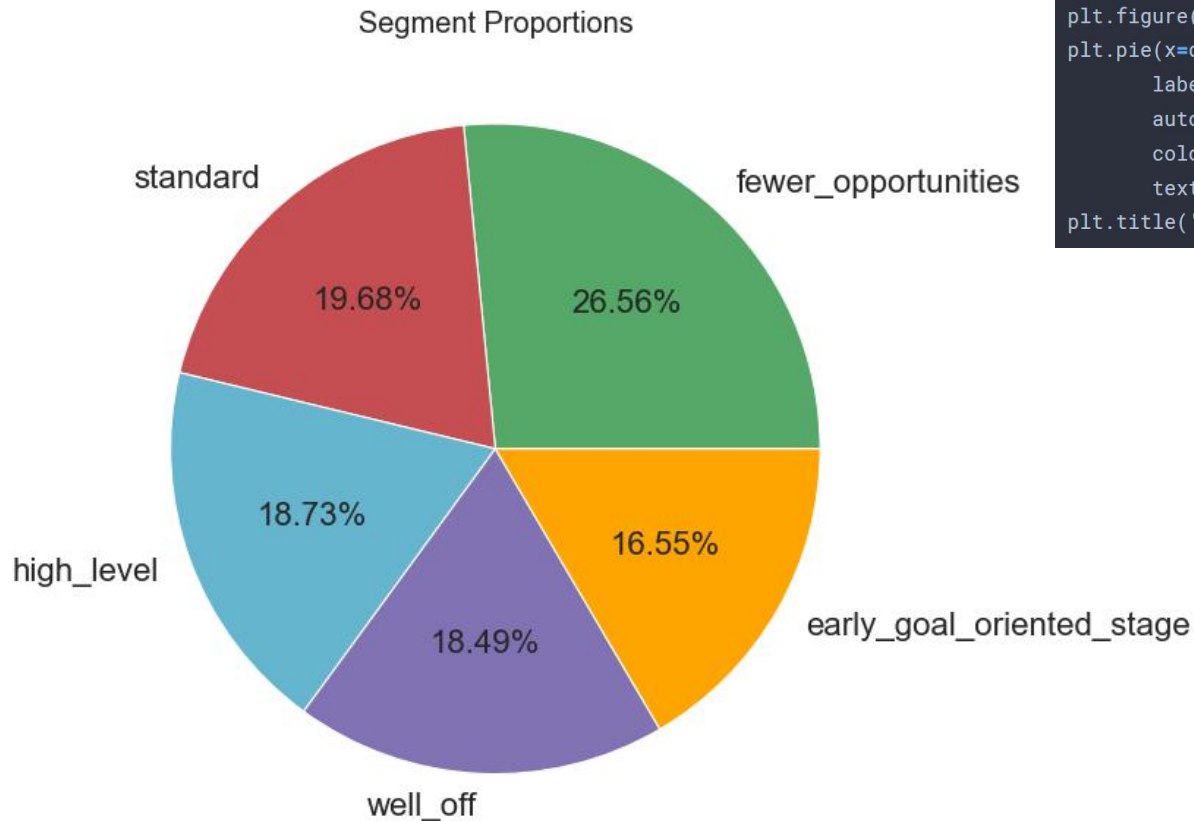
```
features = df_purchase[['Sex', 'Marital status', 'Age', 'Education',  
                        'Income', 'Occupation', 'Settlement size']]  
df_purchase_segm_std = scaler.transform(features)  
df_purchase_segm_pca = pca.transform(df_purchase_segm_std)  
purchase_segm_kmeans_pca = kmeans_pca.predict(df_purchase_segm_pca)  
df_purchase_predictors = df_purchase.copy()  
df_purchase_predictors['Segment'] = purchase_segm_kmeans_pca  
df_purchase_predictors
```

[illegible]

Exploratory Data Analysis

Segments Portions

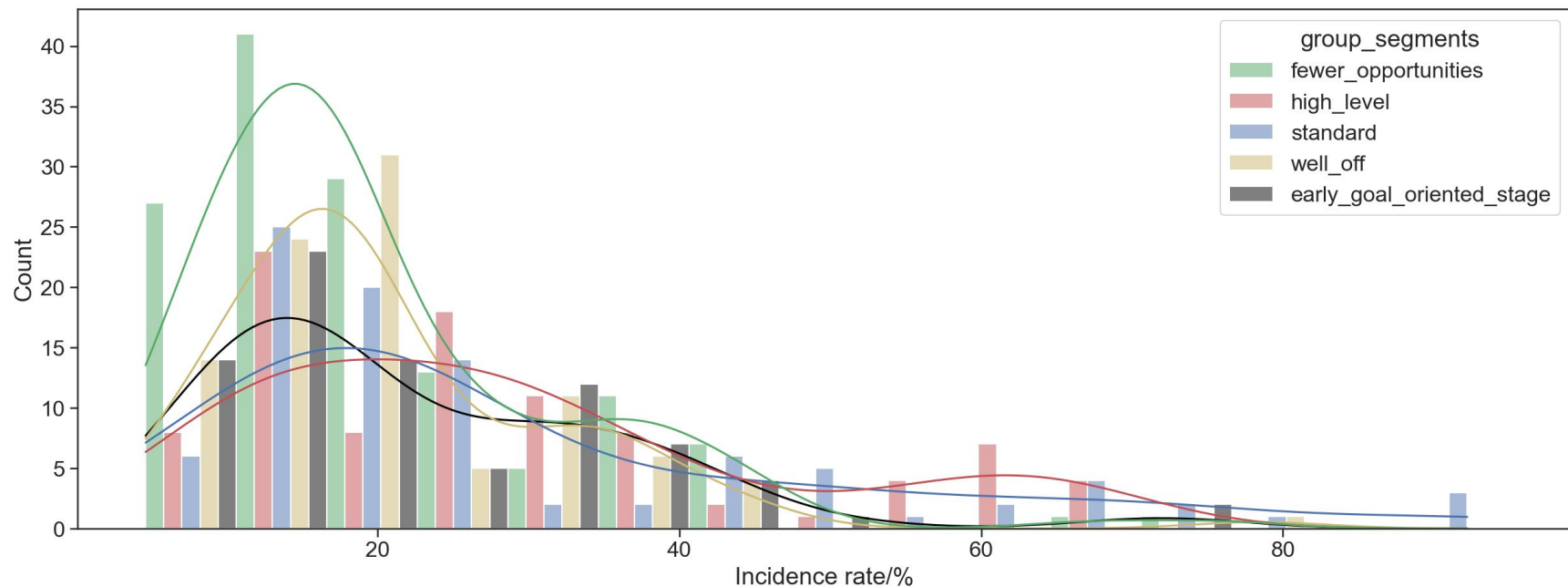
- 圓餅圖可以看到Fewer opportunities的區塊的拜訪購買次數的比例比較大



```
'''了解各區塊的分佈比例'''  
df_group_protion = df_purchase_segkmeans_pca['Segments'].value_counts()/len(df_purchase_segkmeans_pca)  
labels = df_group_protion.rename(({0:'high_level',  
                                   1:'fewer_opportunities',  
                                   2:'well_off',  
                                   3:'early_goal_oriented_stage',  
                                   4:'standard'})).index  
  
plt.figure(figsize = (16,9), dpi=80)  
plt.pie(x=df_group_protion,  
        labels = labels,  
        autopct='%1.2f%%',  
        colors = ['g', 'r', 'c', 'm', 'orange'],  
        textprops={'fontsize':20});  
plt.title('Segment Proportions');
```

Incidence rate

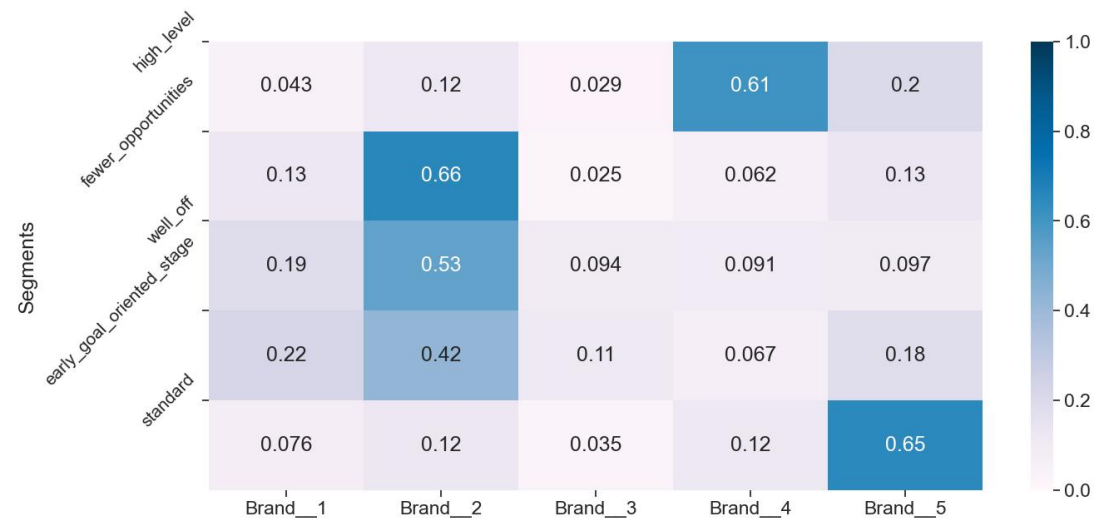
- 用新的Purchase Data裡面的資料計算Incidence rate，即是各個ID每次到該賣場有購買巧克力的機率，這邊可以看到在fewer-opportunity以及well_off的族群，購買率集中在0~30之間，購買機率不大，該產品對於此Segment的消費者來說並非為必需品，相較於其他三個族群來說波動差距比較大。



Exploratory Data Analysis

Segments_Brand

- 預設:Brand_1~_5，1的價格是最低，而5是最高
- 依照segments來看可以理解個別所偏好的品牌價格，右圖可以看到Career-Focused和Well-Off分別對4及5的品牌有所偏好，比較特別的是fewer opportunity的品牌偏好竟然不是價格最低的，所以在定價策略上面最低價絕對不是最好的選擇。



```
#把brand攤開成欄的方式去檢視
df_purchase_incidence = df_purchase_predictors[df_purchase_predictors['Incidence'] == 1]
brand_dummies = pd.get_dummies(df_purchase_incidence['Brand'], prefix = 'Brand', prefix_sep = '_')
brand_dummies['Segment'], brand_dummies['ID'] = df_purchase_incidence['Segment'], df_purchase_incidence['ID']
#檢視Segments各Brand的平均購買比例
temp = brand_dummies.groupby(['ID'], as_index = True).mean()
mean_brand_choice = temp.groupby(['Segment'], as_index = True).mean()

sns.heatmap(mean_brand_choice,
            vmin = 0,
            vmax = 1,
            cmap = 'PuBu',
            annot = True)

plt.yticks([0, 1, 2, 3], ['Standard', 'Career-Focused', 'Fewer-Opportunities', 'Well-Off'], rotation = 45, fontsize = 9)
plt.title('Average Brand Choice by Segment')
plt.show()
```


Price elasticity formula

公式為數量的變動對於價格的變動比例，一般如果 $|E| < 1$ ，都歸類為inelastic，簡單來說當財貨價格產生變動時，對需求的影響較小。

經過數學公式轉換求得下面公式，得以利用Python計算：

$$\text{Elasticity} \leftarrow E = (1 - Y) * \beta_1 * P \rightarrow \text{Price}$$

Logistic regression coefficient of Price (points to β_1)

Purchase probability (points to Y)

Calculating Percentage Changes

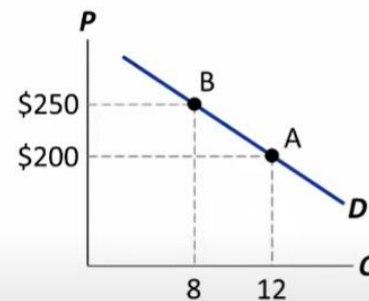
Using the midpoint method:

$$\% \text{ change in } Q = \frac{12 - 8}{10} \times 100\% = 40.0\%$$

$$\% \text{ change in } P \text{ equals } \frac{\$250 - \$200}{\$225} \times 100\% = 22.2\%$$

The price elasticity of demand equals

$$40/22.2 = 1.8$$



右圖片來源：<https://www.youtube.com/watch?v=5UKw4blQdsc>

Price Elasticity of Purchase Probability Coefficient

利用LogisticRegression先求出相關係數(Coefficient)，該資料裡的所有價格做加總再平均後得出X，拿平均價格(X)和發生購買事件(Y)做訓練求得相關係數

```
#使用原資料df_purchase_predictors改名為df_pa
df_pa = df_purchase_predictors
#要預測purchase probability，故把Incidence設為Y，把所有價格平均設為X
Y = df_pa['Incidence']
df_purchase_predictors
X = pd.DataFrame()
X['Mean_Price'] = (df_pa['Price_1'] +
                  df_pa['Price_2'] +
                  df_pa['Price_3'] +
                  df_pa['Price_4'] +
                  df_pa['Price_5']) / 5

#使用Logistic Regression模型，fit模型(X,Y)
model_purchase = LogisticRegression(solver = 'sag')
model_purchase.fit(X, Y)
model_purchase.coef_

array([[ -2.34816446]])
```

Price Elasticity of Purchase Probability

Average Price

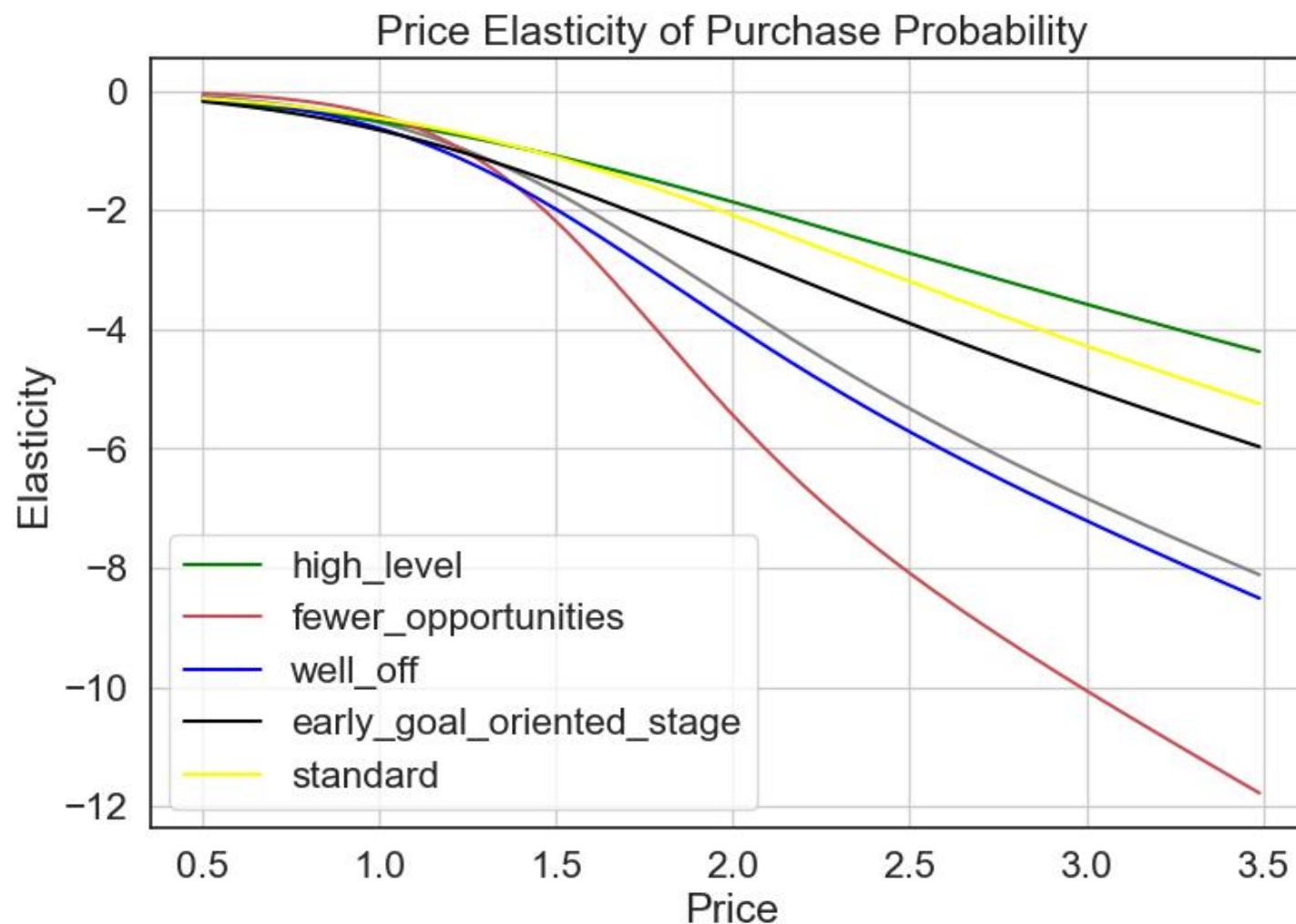
```
price_range = np.arange(0.5, 3.5, 0.01)
#算出LogisticRegression的predict_proba算出Purchase Probability
df_price_range = pd.DataFrame(price_range)
Y_pr = model_purchase.predict_proba(df_price_range)
purchase_pr = Y_pr[:, 1]
#Elasticity 公式  $E = (1-Y)*B*P$ 
pe = model_purchase.coef[:, 0] * price_range * (1 - purchase_pr)
#
plt.figure(figsize = (9, 6))
plt.plot(price_range, pe, color = 'grey')
plt.xlabel('Price')
plt.ylabel('Elasticity')
plt.title('Price Elasticity of Purchase Probability')
```



上圖可以看到，價格彈性跟價格是負相關的。在差不多價格1.3元之後，消費者對於商品價格將會比較強烈的回應。

Purchase Probability by Segments

利用價格彈性公式可以看到不同族群的人對於價格有所不同的反應，其中紅色線是最明顯的，在約1.5元之後就呈現急速下降，其次是藍色線，也就是我們的 **well_off** 族群，可以想像，因為先前已經看到這兩個族群對該品牌並沒有特別偏好，故價格對他們的影響也會比其他三個族群高。



PySpark & scikit-learn

利用隨機樹演算法進行網格搜索以及交叉驗證，針對消費者購買的品牌做預測，分數(**accuracy**)落在**0.65**附近。

訓練參數包含：

1. 該消費者特徵分群
2. 各牌子的價格
3. 各牌子是否有促銷
4. 第幾天到訪

共**12**種

總結:

- 這次的主要分析為:
 - 分類 - 運用三種模型可以有效進行分類。
 - 品牌的選擇 - 運用LogisticRegression搭配heatmap的mean_brand_choice了解到顧客偏好以及對價格的反應程度。
 - 使用PySpark以及sickit-learn對該資料預測分數並不高，有可能是給予的特徵不足以判斷或是資料量太少。
- 經過這次的分析，對於顧客的掌握度提高，更加瞭解目標客群，針對不同客群做出不同對策，對於公司未來產品發展會有明顯的幫助，至於預測結果可以蒐集更多特徵優化預測結果。