pincode-analysis-clustering-analysis-kmeans

September 5, 2024

0.1 OBJECTIVE

Clustering Analysis:

Pincode Analysis excel contains data related to ecommerce site. First column is pincode in encrypted format, which you can replace with valid 6 digit numeric pincodes. Each row is unique on Pincode, so you need to add unique pin codes in each of the rows. Last columns total_orders is the number of orders placed/delivered in the corresponding pincode in a given time frame. All other columns are various categories of products sold in these orders. Numbers in each of these categories are %of total orders, so if you add up all the columns in a row, except first(pincode) and last(total_orders) it adds up to 100%. The task is to find if any categories are dominant in any pincode, or across few pin codes or all categories are spread uniformly across all pin codes. So business problem is to figure out if there is any category affinity towards any geographical area(pincode).

[1]:
```
! pip install kneed
import pandas as pd
import random
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.decomposition import PCA
import numpy as np
from sklearn.cluster import KMeans
from kneed import KneeLocator
import joblib
from sklearn.metrics import silhouette_score
from geopy.geocoders import Nominatim
```

```
Collecting kneed
  Downloading kneed-0.8.5-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: numpy>=1.14.2 in /opt/conda/lib/python3.10/site-
packages (from kneed) (1.26.4)
Requirement already satisfied: scipy>=1.0.0 in /opt/conda/lib/python3.10/site-
packages (from kneed) (1.11.4)
Downloading kneed-0.8.5-py3-none-any.whl (10 kB)
Installing collected packages: kneed
Successfully installed kneed-0.8.5
```

0.1.1 DATA EXPLORATION

[2]:
```
df = pd.read_excel('/kaggle/input/pincode-analysis-xlsx/Pincode Analysis.xlsx')
df
```

[2]:

	DAC49DE9MyETlKVkFxtZrPJu26RqezfY7+JccDvV	Agriculture	Appliances	\
0	85A6B3EEMyETlKVkFxuy1z709sCcgQ==	NaN	0.446789	
1	914EC099MyETlKVkFxtbgaaUnW5FQw==	NaN	0.129535	
2	87404181MyETlKVkFxs0XoE03yRciA==	NaN	0.496763	
3	F9D55206MyETlKVkFxtrzbWIZlmuvg==	0.001344	0.931189	
4	135A7354MyETlKVkFxtAzxP199AV9g==	NaN	0.473075	
..	…	…	…	
142	4BB912A3MyETlKVkFxv4HxtbbibB1Q==	NaN	2.043040	
143	1AD03B94MyETlKVkFxuvbDV/UcKmqw==	NaN	0.404526	
144	1D20DDE3MyETlKVkFxthlSEtwisn/Q==	NaN	0.936012	
145	580D9354MyETlKVkFxtyRg1ZhQ5KnA==	0.126186	4.822516	
146	9374B270MyETlKVkFxurCmZSiRpcJg==	NaN	2.174395	

	BPC	Electronics	F&B	Fashion	Grocery	\
0	0.513025	0.210618	9.234456	2.274875	85.610235	
1	0.126827	0.064993	0.115543	0.487900	98.737148	
2	0.847896	0.236457	72.982504	2.554345	20.691510	
3	1.182462	0.487766	49.773585	5.216941	38.597976	
4	0.820798	0.200907	57.399203	3.096882	35.344976	
..	…	…	…	…	…	
142	2.402615	0.915282	69.125579	9.169164	9.267230	
143	0.634122	0.235063	20.543377	3.083147	72.322746	
144	1.138541	0.470743	29.032788	4.280475	60.966665	
145	6.814067	2.084819	3.818511	19.388819	46.913919	
146	2.202130	0.931884	51.442201	10.761038	24.667184	

	Health & Wellness	Home & Kitchen	Multi Category	Others	Undefined	\
0	0.471348	1.165471	0.018606	0.029521	0.025056	
1	0.117800	0.209874	0.004062	0.004965	0.001354	
2	0.590128	1.382208	0.182163	0.025878	0.010148	
3	1.011139	2.650462	0.049045	0.054420	0.043670	
4	0.613023	1.744625	0.261007	0.018030	0.027474	
..	…	…	…	…	…	
142	1.765187	5.088532	0.103514	0.087170	0.032689	
143	0.453725	2.279560	0.021866	0.021866	NaN	
144	0.738957	2.238765	0.076633	0.098528	0.021895	
145	4.147693	11.499424	0.148132	0.230427	0.005486	
146	2.435101	5.075438	0.166408	0.133126	0.011094	

	total_orders
0	403099
1	221562
2	197076

```
3          148842
4          116472
..            …
142         18355
143         18293
144         18269
145         18227
146         18028

[147 rows x 14 columns]
```

[3]:
```
# Replacing with valid 6 digit numeric pincodes
df.rename(columns={'DAC49DE9MyETlKVkFxtZrPJu26RqezfY7+JccDvV': 'pincode'},␣
 ↪inplace=True)
random.seed(100)
generated_pins = set()
def generate_unique_pin():
    while True:
        pin = random.randint(111111, 999999)
        if pin not in generated_pins:
            generated_pins.add(pin)
            return pin

df['pincode'] = df['pincode'].apply(lambda pin: random.randint(111111,999999))
df
```

[3]:

	pincode	Agriculture	Appliances	BPC	Electronics	F&B
0	263856	NaN	0.446789	0.513025	0.210618	9.234456
1	592961	NaN	0.129535	0.126827	0.064993	0.115543
2	588136	NaN	0.496763	0.847896	0.236457	72.982504
3	919336	0.001344	0.931189	1.182462	0.487766	49.773585
4	294347	NaN	0.473075	0.820798	0.200907	57.399203
..	…	…	…	…	…	…
142	609726	NaN	2.043040	2.402615	0.915282	69.125579
143	313306	NaN	0.404526	0.634122	0.235063	20.543377
144	973109	NaN	0.936012	1.138541	0.470743	29.032788
145	661910	0.126186	4.822516	6.814067	2.084819	3.818511
146	235981	NaN	2.174395	2.202130	0.931884	51.442201

	Fashion	Grocery	Health & Wellness	Home & Kitchen	Multi Category
0	2.274875	85.610235	0.471348	1.165471	0.018606
1	0.487900	98.737148	0.117800	0.209874	0.004062
2	2.554345	20.691510	0.590128	1.382208	0.182163
3	5.216941	38.597976	1.011139	2.650462	0.049045
4	3.096882	35.344976	0.613023	1.744625	0.261007
..	…	…	…	…	…
142	9.169164	9.267230	1.765187	5.088532	0.103514

```
143   3.083147  72.322746           0.453725        2.279560        0.021866
144   4.280475  60.966665           0.738957        2.238765        0.076633
145  19.388819  46.913919           4.147693       11.499424        0.148132
146  10.761038  24.667184           2.435101        5.075438        0.166408

       Others  Undefined  total_orders
0    0.029521   0.025056        403099
1    0.004965   0.001354        221562
2    0.025878   0.010148        197076
3    0.054420   0.043670        148842
4    0.018030   0.027474        116472
..        …        …             …
142  0.087170   0.032689         18355
143  0.021866        NaN         18293
144  0.098528   0.021895         18269
145  0.230427   0.005486         18227
146  0.133126   0.011094         18028

[147 rows x 14 columns]
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 147 entries, 0 to 146
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   pincode            147 non-null    int64
 1   Agriculture        5 non-null      float64
 2   Appliances         147 non-null    float64
 3   BPC                147 non-null    float64
 4   Electronics        147 non-null    float64
 5   F&B                147 non-null    float64
 6   Fashion            147 non-null    float64
 7   Grocery            147 non-null    float64
 8   Health & Wellness  147 non-null    float64
 9   Home & Kitchen     147 non-null    float64
 10  Multi Category     142 non-null    float64
 11  Others             147 non-null    float64
 12  Undefined          133 non-null    float64
 13  total_orders       147 non-null    int64
dtypes: float64(12), int64(2)
memory usage: 16.2 KB
```

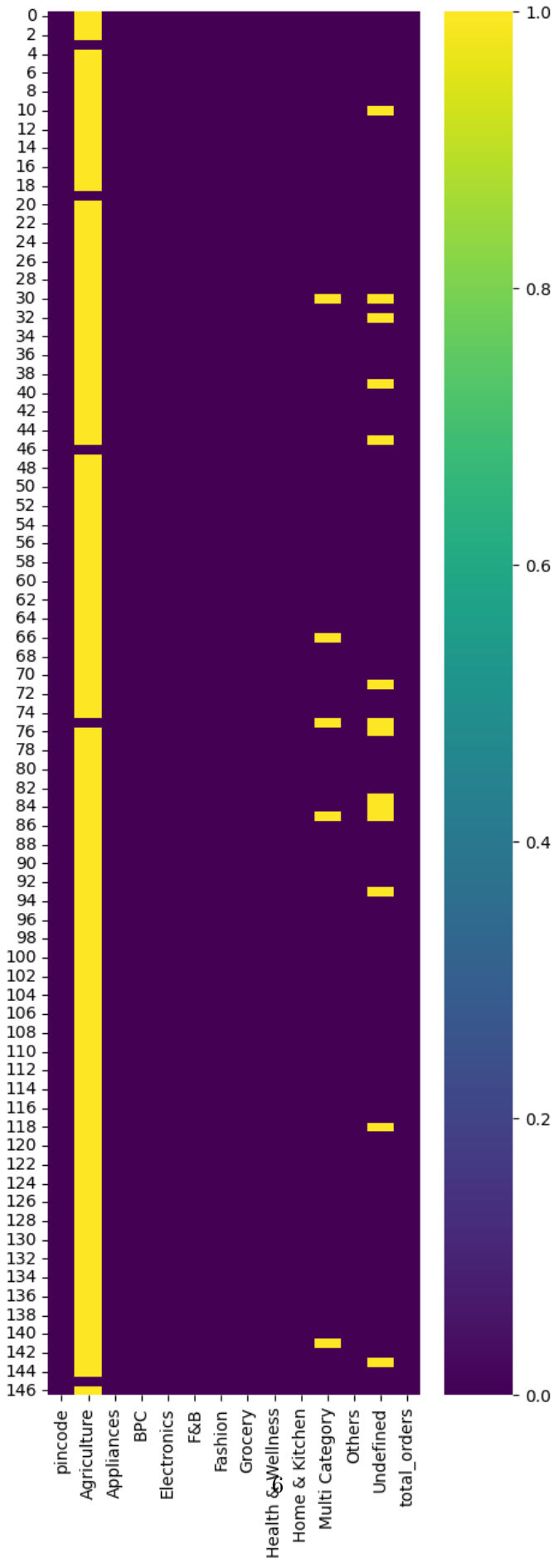
0.2 Data Cleaning

[5]:
```
#Checking for null values
df.isnull().sum()
```

[5]:
```
pincode               0
Agriculture         142
Appliances            0
BPC                   0
Electronics           0
F&B                   0
Fashion               0
Grocery               0
Health & Wellness     0
Home & Kitchen        0
Multi Category        5
Others                0
Undefined            14
total_orders          0
dtype: int64
```

[6]:
```
#Plotting heatmap  to visulize the missing values
plt.figure(figsize=(5, 15))
sns.heatmap(df.isnull(),cmap = 'viridis')
```

[6]:
```
<Axes: >
```


0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62
64
66
68
70
72
74
76
78
80
82
84
86
88
90
92
94
96
98
100
102
104
106
108
110
112
114
116
118
120
122
124
126
128
130
132
134
136
138
140
142
144
146
pincode
Agriculture
Appliances
BPC
Electronics
F&B
Fashion
Grocery
Health & Wellness
Home & Kitchen
Multi Category
Others
Undefined
total_orders
1.0
0.8
0.6
0.4
0.2
0.0

[7]:
```
# Imputing Null values in 'Agriculture' with 0
df['Agriculture'] = df['Agriculture'].fillna(0)
df['Agriculture']
```

[7]:
```
0      0.000000
1      0.000000
2      0.000000
3      0.001344
4      0.000000
         …
142    0.000000
143    0.000000
144    0.000000
145    0.126186
146    0.000000
Name: Agriculture, Length: 147, dtype: float64
```

[7]:
```
0      0.000000
1      0.000000
2      0.000000
3      0.001344
4      0.000000
         …
142    0.000000
143    0.000000
144    0.000000
145    0.126186
146    0.000000
Name: Agriculture, Length: 147, dtype: float64
```

[8]:
```
# Removing rows where Multi Category & Undefined are None
df = df[~(df['Multi Category'].isna() & df['Undefined'].isna())]
df.reset_index(drop=True,inplace = True)
df
```

[8]:
```
     pincode  Agriculture  Appliances       BPC  Electronics        F&B  \
0     263856     0.000000    0.446789  0.513025     0.210618   9.234456
1     592961     0.000000    0.129535  0.126827     0.064993   0.115543
2     588136     0.000000    0.496763  0.847896     0.236457  72.982504
3     919336     0.001344    0.931189  1.182462     0.487766  49.773585
4     294347     0.000000    0.473075  0.820798     0.200907  57.399203
..       …          …          …        …            …          …
139   609726     0.000000    2.043040  2.402615     0.915282  69.125579
140   313306     0.000000    0.404526  0.634122     0.235063  20.543377
141   973109     0.000000    0.936012  1.138541     0.470743  29.032788
```

```
142   661910     0.126186    4.822516  6.814067     2.084819   3.818511
143   235981     0.000000    2.174395  2.202130     0.931884  51.442201

       Fashion    Grocery  Health & Wellness  Home & Kitchen  Multi Category  \
0     2.274875  85.610235           0.471348        1.165471        0.018606
1     0.487900  98.737148           0.117800        0.209874        0.004062
2     2.554345  20.691510           0.590128        1.382208        0.182163
3     5.216941  38.597976           1.011139        2.650462        0.049045
4     3.096882  35.344976           0.613023        1.744625        0.261007
..         …         …                  …               …               …
139   9.169164   9.267230           1.765187        5.088532        0.103514
140   3.083147  72.322746           0.453725        2.279560        0.021866
141   4.280475  60.966665           0.738957        2.238765        0.076633
142  19.388819  46.913919           4.147693       11.499424        0.148132
143  10.761038  24.667184           2.435101        5.075438        0.166408

       Others  Undefined  total_orders
0    0.029521   0.025056        403099
1    0.004965   0.001354        221562
2    0.025878   0.010148        197076
3    0.054420   0.043670        148842
4    0.018030   0.027474        116472
..        …          …             …
139  0.087170   0.032689         18355
140  0.021866        NaN         18293
141  0.098528   0.021895         18269
142  0.230427   0.005486         18227
143  0.133126   0.011094         18028

[144 rows x 14 columns]
```

[9]:
```python
# Imputing the Null values in the 'Undefined' and 'Multi Category' with the␣
 ↪difference between 100% and the sum of the other known values in that row.
def impute(row, col):
    if pd.isna(row[col]):
        known_sum = row.drop([col]).sum()
        missing_value = 100 - known_sum
        return max(0, min(100, missing_value))  # to ensure value is within [0,␣
 ↪100]
    else:
        return row[col]

for col in ['Multi Category', 'Undefined']:
    df.loc[:,col] = df.apply(lambda row: impute(row, col), axis=1)
df
```

[9]:

	pincode	Agriculture	Appliances	BPC	Electronics	F&B	\
0	263856	0.000000	0.446789	0.513025	0.210618	9.234456	
1	592961	0.000000	0.129535	0.126827	0.064993	0.115543	
2	588136	0.000000	0.496763	0.847896	0.236457	72.982504	
3	919336	0.001344	0.931189	1.182462	0.487766	49.773585	
4	294347	0.000000	0.473075	0.820798	0.200907	57.399203	
..	…	…	…	…	…	…	
139	609726	0.000000	2.043040	2.402615	0.915282	69.125579	
140	313306	0.000000	0.404526	0.634122	0.235063	20.543377	
141	973109	0.000000	0.936012	1.138541	0.470743	29.032788	
142	661910	0.126186	4.822516	6.814067	2.084819	3.818511	
143	235981	0.000000	2.174395	2.202130	0.931884	51.442201	

	Fashion	Grocery	Health & Wellness	Home & Kitchen	Multi Category	\
0	2.274875	85.610235	0.471348	1.165471	0.018606	
1	0.487900	98.737148	0.117800	0.209874	0.004062	
2	2.554345	20.691510	0.590128	1.382208	0.182163	
3	5.216941	38.597976	1.011139	2.650462	0.049045	
4	3.096882	35.344976	0.613023	1.744625	0.261007	
..	…	…	…	…	…	
139	9.169164	9.267230	1.765187	5.088532	0.103514	
140	3.083147	72.322746	0.453725	2.279560	0.021866	
141	4.280475	60.966665	0.738957	2.238765	0.076633	
142	19.388819	46.913919	4.147693	11.499424	0.148132	
143	10.761038	24.667184	2.435101	5.075438	0.166408	

	Others	Undefined	total_orders
0	0.029521	0.025056	403099
1	0.004965	0.001354	221562
2	0.025878	0.010148	197076
3	0.054420	0.043670	148842
4	0.018030	0.027474	116472
..	…	…	…
139	0.087170	0.032689	18355
140	0.021866	0.000000	18293
141	0.098528	0.021895	18269
142	0.230427	0.005486	18227
143	0.133126	0.011094	18028

[144 rows x 14 columns]

[10]:
```
#check for remaining null values
df.isnull().sum()
```

[10]:
```
pincode             0
Agriculture         0
Appliances          0
```

```
BPC                  0
Electronics          0
F&B                  0
Fashion              0
Grocery              0
Health & Wellness    0
Home & Kitchen       0
Multi Category       0
Others               0
Undefined            0
total_orders         0
dtype: int64
```

[11]:
```
#check if the sum in each row is 100
((round(df.iloc[:,1:-1].sum(axis=1),1) < 100.0) | (round(df.iloc[:,1:-1].
 ↪sum(axis=1),1) > 100.0)) .sum()
```

[11]:
```
0
```

[12]:
```
# Summary statistics
df.iloc[:,1:].describe()
```

[12]:

	Agriculture	Appliances	BPC	Electronics	F&B
count	144.000000	144.000000	144.000000	144.000000	144.000000
mean	0.001017	1.332071	1.606208	0.575435	57.817697
std	0.010593	1.561655	1.670165	0.601656	26.387076
min	0.000000	0.065035	0.126827	0.031904	0.005119
25%	0.000000	0.565619	0.777530	0.254610	40.379280
50%	0.000000	0.944460	1.146697	0.426418	63.943781
75%	0.000000	1.401625	1.748100	0.643823	78.081153
max	0.126186	10.852316	11.430117	5.273910	95.311801

	Fashion	Grocery	Health & Wellness	Home & Kitchen
count	144.000000	144.000000	144.000000	144.000000
mean	6.215421	27.357465	1.364283	3.504570
std	5.751996	24.166147	1.447606	3.830727
min	0.487900	0.421348	0.093257	0.201239
25%	3.219614	6.069089	0.625824	1.762271
50%	4.994577	20.846359	0.954048	2.688859
75%	7.260319	40.716283	1.521809	3.783161
max	37.997914	98.737148	9.399735	29.065779

	Multi Category	Others	Undefined	total_orders
count	144.000000	144.000000	144.000000	144.000000
mean	0.130113	0.070428	0.025291	40837.402778
std	0.364984	0.110250	0.021145	42555.415644
min	0.000000	0.004965	0.000000	18028.000000

```
25%          0.035641    0.028470    0.009603   22128.250000
50%          0.069844    0.047152    0.021317   28311.000000
75%          0.113086    0.074918    0.034883   41025.500000
max          4.175103    0.888018    0.115047  403099.000000
```

[13]:
```
# Checking for collinearity
pd.Series([variance_inflation_factor(df.iloc[:,1:-1], i) for i in range(df.
  ↪iloc[:,1:-1].shape[1])], index = df.columns[1:-1])
```

[13]:
```
Agriculture           1.263064
Appliances          104.827459
BPC                  50.220310
Electronics          59.034247
F&B                   2.594706
Fashion             115.385178
Grocery               1.896131
Health & Wellness     9.578339
Home & Kitchen      135.663626
Multi Category        1.202097
Others                4.865206
Undefined             2.624882
dtype: float64
```

[14]:
```
# Since there is inherent mullticollinearity, using PCA to address it
pca = PCA(n_components=2)
principal_components = pca.fit_transform(df.iloc[:,1:-1])
print(f'explained_variance_ratio : {pca.explained_variance_ratio_}')
print(f'Cumulative explained_variance_ratio: {pca.explained_variance_ratio_.
  ↪sum()}')
print('principal component:')
principal_components
```

```
explained_variance_ratio : [0.88496395 0.11334879]
Cumulative explained_variance_ratio: 0.9983127380312116
principal component:
```

[14]:
```
array([[ 74.82303257, -13.37948455],
       [ 90.28537451, -17.94478844],
       [-15.89682436,  -6.43821256],
       [ 13.42805971,  -3.58351712],
       [  5.49513214,  -6.96786829],
       [-42.21316112,  -6.81326243],
       [-23.61886705,  -3.37727088],
       [-11.45512246,  -4.68762056],
       [-35.49157223,  -2.89124535],
       [-37.33644969,  -4.62258802],
       [ 89.43030554, -17.54968202],
       [ 10.46850076,  -7.01358145],
```

```
[-30.58298276,  -6.45114615],
[-38.06333815,  -6.39870364],
[-35.47979597,  -4.00798827],
[-43.90119332,  -7.01545948],
[ 25.49181294,  -7.46411659],
[ 67.62282114, -13.4897842 ],
[-16.97498778,  -4.93856834],
[ 37.61159454,  -9.90370196],
[-40.69456443,  -5.73903044],
[-40.42236582,  -6.54322792],
[ 33.67283144,  -5.85357105],
[-40.93507037,  -2.99568663],
[ 44.5556693 ,  -5.42165625],
[ 17.01310797,   1.45694915],
[-29.22212733,  -4.50702268],
[  4.98944411,  -3.48207762],
[-35.00408014,   0.32081523],
[ 11.27115467,  -7.30641498],
[ 55.28257233,  -2.85392164],
[-42.86093486,  -2.72645561],
[-37.25344534,  -3.43291426],
[ 43.45476083, -10.22870738],
[  2.32051845,  -0.7730495 ],
[ 27.95957116,  -6.30415163],
[-30.65700224,   3.38985143],
[-39.4122972 ,   0.14559448],
[ -7.18642197,   2.49666837],
[  5.32796173,  -7.68275109],
[-17.98296067,  -5.68024213],
[-21.37053903,  -1.36512508],
[-29.62020985,  -2.56565683],
[-14.5045803 ,  -1.26456515],
[-25.88181694,  -4.02495941],
[ 18.40755177, -10.49017464],
[-36.36970212,  -4.09218511],
[-33.27560124,  -5.95657409],
[-40.12872173,  -4.51691005],
[ 25.21987747,  -8.63846749],
[ 45.38659761,   2.733867  ],
[ 42.81230291,  -7.81918052],
[-19.85279885,  -3.01524067],
[-44.28634568,  -6.38956141],
[ 58.93228736,  -7.85144666],
[  0.60097289,   0.41309041],
[-12.29267379,  -1.48419177],
[ 55.09730284,  -0.80853596],
[ 14.02074063,  -0.4493042 ],
```

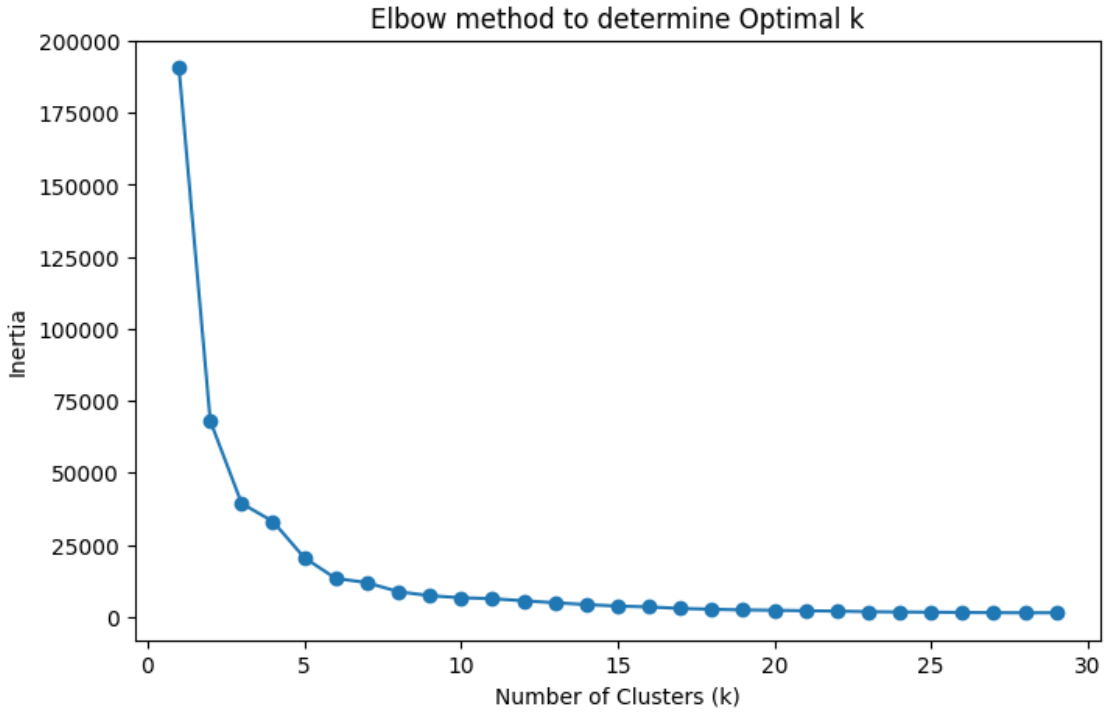
```
[-35.61714514,   3.44605156],
[-11.79662444,  -6.1123024 ],
[-33.35576691,  -3.97459881],
[-25.81767927,   0.25452145],
[-18.37161603,   7.7675264 ],
[-29.11508238,   3.82377361],
[-13.75975298,  -4.80554437],
[ 67.88935758, -10.41092039],
[ -9.29741176,  -0.59658103],
[-11.24044333,   0.21542366],
[ 29.87295838,  -8.72150872],
[-18.05489322,  10.2703113 ],
[ 53.3910021 ,  -2.92786875],
[ 58.74198022,  -1.74034161],
[-42.72565172,  -3.50077667],
[ 28.82828693, -10.34942646],
[ 42.72866162,  -6.29196398],
[-20.57479252,  -6.08636033],
[ 26.08966038,   0.60604325],
[-45.69733929,  -6.68551137],
[ 34.41025951,  -8.38248268],
[  6.66567482,   1.672268  ],
[-18.63510613,  -7.49431059],
[  7.99004476,   0.88734862],
[-20.20647962,   2.65377544],
[-36.84045001,   0.31471596],
[  6.33650936,   2.46703039],
[-40.0496505 ,   0.47663818],
[  1.94120051,  16.03368568],
[ 73.26664727,  -6.95061042],
[  3.00037022,   4.54459211],
[ -8.37171316,   2.86952354],
[-17.2059816 ,  -1.95301418],
[ 12.75366727,   3.99561959],
[ -6.07323698,  -0.79157458],
[ 67.26534022,  11.72786038],
[ -1.59513182,   0.45092327],
[ -4.96433203,   7.923499  ],
[ 21.38060669,   4.95637473],
[-42.76751021,  -3.92763642],
[ -7.47552927,   0.98149983],
[ 60.20540447,  -0.44308252],
[ 46.78713017,  -3.62646457],
[-26.78391584,   0.50217934],
[-25.64035775,   4.99398326],
[-12.6304037 ,  -6.74839387],
[  3.02654352,  -8.66760824],
```

```
       [-31.03817817,   0.93067668],
       [ 39.1322402 ,  -3.38249871],
       [ -0.41981904,  -2.03510917],
       [-30.07254877,   9.24214768],
       [  5.20957877,   5.15706542],
       [ 53.71167748,  17.33238182],
       [-24.6951804 ,   3.14518461],
       [-17.88516234,  -0.19221083],
       [-34.90798762,   5.23244544],
       [-32.56771125,   6.83923673],
       [-24.74080378,   8.26902762],
       [-40.22822812,  -2.88941604],
       [-14.052024  ,  -5.06804663],
       [  6.97381929,   6.94437641],
       [ 25.05651608,  66.40745133],
       [-12.542193  ,  -4.01271939],
       [  7.35680919,  -2.24799927],
       [-34.92868503,  -3.25323323],
       [  7.88696258,  13.14584272],
       [ 15.97233135,  13.3600466 ],
       [ 28.64562212,  68.99722716],
       [ 26.62341865,  -4.54989826],
       [-41.84337989,  -4.49943376],
       [ 22.81569411,  64.33597288],
       [-24.46462957,  -2.71949127],
       [-37.18573432,   0.65991842],
       [-14.58715033,  -2.39426598],
       [  6.94726507,  43.86229963],
       [ 17.71368283,   2.41658811],
       [-21.07362638,  11.24626684],
       [-12.94418672,  -2.66621992],
       [-40.77131278,  -0.36446302],
       [ 70.26581529,  -7.43299854],
       [-20.34801499,   7.31793771],
       [ 57.59095963, -10.38203339],
       [ 43.74450226,  -7.11686075],
       [ 53.85790685,  26.10290223],
       [  3.13114188,   7.86033062]])
```

0.3 Model Training

[15]:
```
#Using Elbow method to determine Optimal k
inertia = []
k_range = range(1, 30)
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42,n_init='auto')
    kmeans.fit(principal_components)
```

```
    inertia.append(kmeans.inertia_)

plt.figure(figsize=(8, 5))
plt.plot(k_range, inertia, marker='o')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow method to determine Optimal k')
plt.show()

kneedle = KneeLocator(k_range, inertia, curve='convex', direction='decreasing')
elbow_point = kneedle.elbow
print(f'Optimal k : {elbow_point}')
```




```
Optimal k : 6
```

0.4 Evaluate Clustering

[16]:
```
# Training the model using Optimal k(Elbow_point)
df = df.copy()
kmeans = KMeans(n_clusters=elbow_point, random_state=42,n_init='auto')
kmeans.fit(principal_components)
joblib.dump(kmeans,'kmeans_ecommerce.pkl')
df.loc[:, 'cluster'] = kmeans.predict(principal_components)
df
```

[16]:

	pincode	Agriculture	Appliances	BPC	Electronics	F&B	\
0	263856	0.000000	0.446789	0.513025	0.210618	9.234456	
1	592961	0.000000	0.129535	0.126827	0.064993	0.115543	
2	588136	0.000000	0.496763	0.847896	0.236457	72.982504	
3	919336	0.001344	0.931189	1.182462	0.487766	49.773585	
4	294347	0.000000	0.473075	0.820798	0.200907	57.399203	
..	…	…	…	…	…	…	
139	609726	0.000000	2.043040	2.402615	0.915282	69.125579	
140	313306	0.000000	0.404526	0.634122	0.235063	20.543377	
141	973109	0.000000	0.936012	1.138541	0.470743	29.032788	
142	661910	0.126186	4.822516	6.814067	2.084819	3.818511	
143	235981	0.000000	2.174395	2.202130	0.931884	51.442201	

	Fashion	Grocery	Health & Wellness	Home & Kitchen	Multi Category	\
0	2.274875	85.610235	0.471348	1.165471	0.018606	
1	0.487900	98.737148	0.117800	0.209874	0.004062	
2	2.554345	20.691510	0.590128	1.382208	0.182163	
3	5.216941	38.597976	1.011139	2.650462	0.049045	
4	3.096882	35.344976	0.613023	1.744625	0.261007	
..	…	…	…	…	…	
139	9.169164	9.267230	1.765187	5.088532	0.103514	
140	3.083147	72.322746	0.453725	2.279560	0.021866	
141	4.280475	60.966665	0.738957	2.238765	0.076633	
142	19.388819	46.913919	4.147693	11.499424	0.148132	
143	10.761038	24.667184	2.435101	5.075438	0.166408	

	Others	Undefined	total_orders	cluster
0	0.029521	0.025056	403099	1
1	0.004965	0.001354	221562	1
2	0.025878	0.010148	197076	5
3	0.054420	0.043670	148842	2
4	0.018030	0.027474	116472	2
..	…	…	…	…
139	0.087170	0.032689	18355	5
140	0.021866	0.000000	18293	3
141	0.098528	0.021895	18269	3
142	0.230427	0.005486	18227	3
143	0.133126	0.011094	18028	2

[144 rows x 15 columns]

[17]:
```
avg_silhouette = silhouette_score(principal_components, df['cluster'])
avg_silhouette
```

[17]: 0.4920746142878147

[18]:
```
cluster_df = df.groupby('cluster')[df.columns[1:-1]].mean()
cluster_df
```

[18]:

cluster	Agriculture	Appliances	BPC	Electronics	F&B	Fashion
0	0.000000	0.779334	0.964135	0.347458	86.616350	4.134111
1	0.000000	0.554985	0.637999	0.256799	7.699506	2.808589
2	0.000124	1.538811	1.671332	0.708276	49.698380	6.532387
3	0.006191	1.423763	1.715613	0.664767	23.925746	6.450498
4	0.000000	7.938959	9.769253	2.377054	10.431000	34.542645
5	0.000000	1.103475	1.448739	0.504890	70.470970	5.542653

cluster	Grocery	Health & Wellness	Home & Kitchen	Multi Category
0	3.889613	0.971192	2.159215	0.081113
1	85.968383	0.550890	1.369772	0.106973
2	34.637446	1.387583	3.563451	0.144956
3	60.388965	1.407813	3.682857	0.235393
4	3.721528	7.710850	23.121841	0.080534
5	16.487769	1.196972	3.031605	0.112446

cluster	Others	Undefined	total_orders
0	0.034232	0.023245	41743.947368
1	0.037023	0.009082	120572.142857
2	0.083476	0.033777	37892.939394
3	0.076019	0.022376	31538.173913
4	0.282030	0.024305	19483.000000
5	0.075649	0.024833	35808.564103

[19]:
```
df.groupby('cluster')['Agriculture'].count()
```

[19]:
```
cluster
0    38
1     7
2    33
3    23
4     4
5    39
Name: Agriculture, dtype: int64
```

[20]:
```
# Category Dominance

for cluster in cluster_df.index:
    max_value = cluster_df.iloc[:,0:-1].loc[cluster].max()
    dominant_category = cluster_df.iloc[:,0:-1].loc[cluster].idxmax()
    if max_value >= 50:
```

```
        print(f'In cluster {cluster}, dominant category is {dominant_category},␣
↪with an average {round(max_value,2)} %')
    else:
        print(f'In cluster {cluster}, major category is {dominant_category},␣
↪with an average {round(max_value,2)} %')
```

```
In cluster 0, dominant category is F&B, with an average 86.62 %
In cluster 1, dominant category is Grocery, with an average 85.97 %
In cluster 2, major category is F&B, with an average 49.7 %
In cluster 3, dominant category is Grocery, with an average 60.39 %
In cluster 4, major category is Fashion, with an average 34.54 %
In cluster 5, dominant category is F&B, with an average 70.47 %
```

0.5 Geographical Affinity

[21]:
```
# geolocator = Nominatim(user_agent="geoapiExercises")
# def get_lat_lon(pin_code):
#     try:
#         location = geolocator.geocode(f"{pin_code}, India")
#         if location:
#             return (location.latitude, location.longitude)
#         else:
#             return (None, None)
#     except Exception as e:
#         print(f"Error geocoding pin code {pin_code}: {e}")
#         return (None, None)

# df[['latitude', 'longitude']] = df['pincode'].apply(lambda pin: pd.
↪Series(get_lat_lon(pin)))
# df
```

[22]:
```
df['region_number'] = df['pincode'].astype(str).str[0].astype(int)
df
```

[22]:

	pincode	Agriculture	Appliances	BPC	Electronics	F&B	\
0	263856	0.000000	0.446789	0.513025	0.210618	9.234456	
1	592961	0.000000	0.129535	0.126827	0.064993	0.115543	
2	588136	0.000000	0.496763	0.847896	0.236457	72.982504	
3	919336	0.001344	0.931189	1.182462	0.487766	49.773585	
4	294347	0.000000	0.473075	0.820798	0.200907	57.399203	
..	…	…	…	…	…	…	
139	609726	0.000000	2.043040	2.402615	0.915282	69.125579	
140	313306	0.000000	0.404526	0.634122	0.235063	20.543377	
141	973109	0.000000	0.936012	1.138541	0.470743	29.032788	
142	661910	0.126186	4.822516	6.814067	2.084819	3.818511	
143	235981	0.000000	2.174395	2.202130	0.931884	51.442201	

```
       Fashion    Grocery  Health & Wellness  Home & Kitchen  Multi Category  \
0     2.274875  85.610235           0.471348        1.165471        0.018606
1     0.487900  98.737148           0.117800        0.209874        0.004062
2     2.554345  20.691510           0.590128        1.382208        0.182163
3     5.216941  38.597976           1.011139        2.650462        0.049045
4     3.096882  35.344976           0.613023        1.744625        0.261007
..         …         …                  …               …               …
139   9.169164   9.267230           1.765187        5.088532        0.103514
140   3.083147  72.322746           0.453725        2.279560        0.021866
141   4.280475  60.966665           0.738957        2.238765        0.076633
142  19.388819  46.913919           4.147693       11.499424        0.148132
143  10.761038  24.667184           2.435101        5.075438        0.166408

       Others  Undefined  total_orders  cluster  region_number
0    0.029521   0.025056        403099        1              2
1    0.004965   0.001354        221562        1              5
2    0.025878   0.010148        197076        5              5
3    0.054420   0.043670        148842        2              9
4    0.018030   0.027474        116472        2              2
..        …          …             …        …              …
139  0.087170   0.032689         18355        5              6
140  0.021866   0.000000         18293        3              3
141  0.098528   0.021895         18269        3              9
142  0.230427   0.005486         18227        3              6
143  0.133126   0.011094         18028        2              2

[144 rows x 16 columns]
```

[23]:
```
#Check the dominat regions in each cluster

count_df = df.groupby(['cluster', 'region_number']).size().
 ↪reset_index(name='region_count')
max_count_df = count_df.loc[count_df.groupby('cluster')['region_number'].
 ↪idxmax()]
max_count_df.reset_index(inplace = True, drop=True)
max_count_df['total_pincodes'] = df.groupby('cluster').count()['pincode']
print(max_count_df)
print('\n')
for index, row in max_count_df.iterrows():
    print(f"In cluster {row['cluster']}, region {row['region_number']} has the␣
 ↪maximum occurences")
```

```
   cluster  region_number  region_count  total_pincodes
0        0              9             4              38
1        1              6             1               7
2        2              9             8              33
3        3              9             4              23
4        4              9             1               4
```

```
5        5              9              5              39


In cluster 0, region 9 has the maximum occurences
In cluster 1, region 6 has the maximum occurences
In cluster 2, region 9 has the maximum occurences
In cluster 3, region 9 has the maximum occurences
In cluster 4, region 9 has the maximum occurences
In cluster 5, region 9 has the maximum occurences
```

0.6 Region data

1 - Northern Region

Includes states like Delhi, Haryana, Punjab, Himachal Pradesh, and Jammu & Kashmir.

2 - Western Region

Includes states like Rajasthan and Gujarat.

3 - Southern Region

Includes states like Maharashtra and parts of Madhya Pradesh.

4 - Central Region

Includes states like Uttar Pradesh and parts of Madhya Pradesh.

5 - Eastern Region

Includes states like Bihar, Jharkhand, and West Bengal.

6 - North-Eastern Region

Includes states like Assam, Arunachal Pradesh, Nagaland, and others in the northeast.

7 - Southern Region

Includes states like Andhra Pradesh, Telangana, and parts of Tamil Nadu.

8 - Southern Region

Includes states like Tamil Nadu and parts of Karnataka.

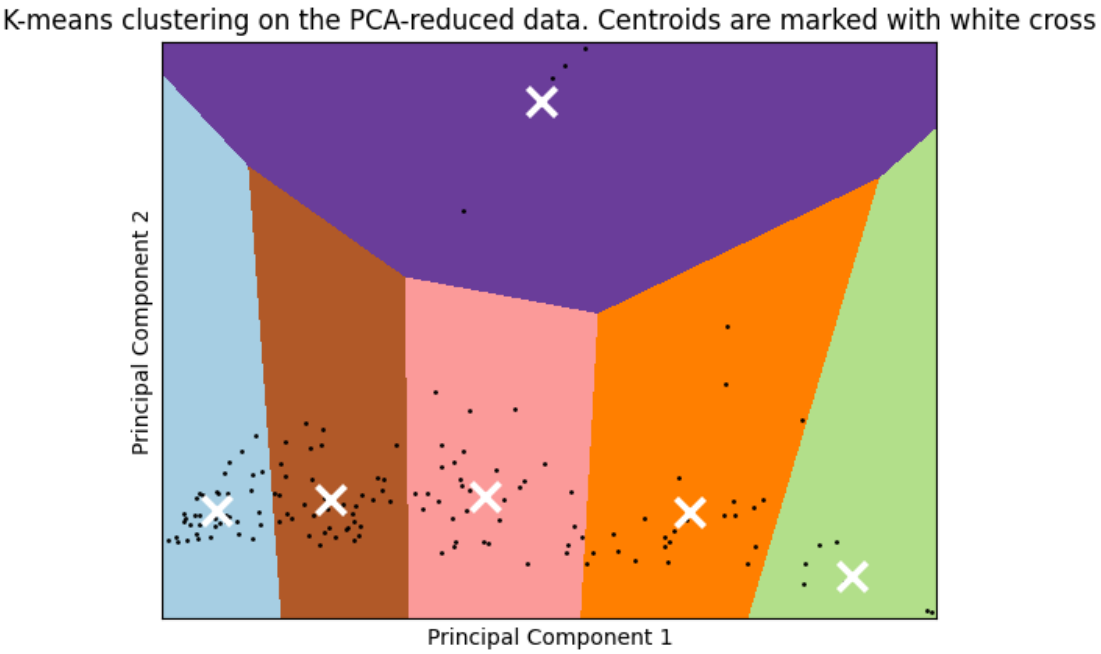
9 - The Army Postal Service (APS) and other specialized regions.

0.7 Visualize the results

[24]:
```
# Step size of the mesh. Decrease to increase the quality of the VQ.
h = 0.02  # point in the mesh [x_min, x_max]x[y_min, y_max].

# Plot the decision boundary. For that, we will assign a color to each
x_min, x_max = principal_components[:, 0].min() - 1, principal_components[:, 0].
 ↪max() + 1
y_min, y_max = principal_components[:, 1].min() - 1, principal_components[:, 1].
 ↪max() + 1
```

```
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

# Obtain labels for each point in mesh. Use last trained model.
Z = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(1)
plt.clf()
plt.imshow(
    Z,
    interpolation="nearest",
    extent=(xx.min(), xx.max(), yy.min(), yy.max()),
    cmap=plt.cm.Paired,
    aspect="auto",
    origin="lower",
)

plt.plot(principal_components[:, 0], principal_components[:, 1], "k.",␣
 ↪markersize=2)

# Plot the centroids as a white X
centroids = kmeans.cluster_centers_
plt.scatter(
    centroids[:, 0],
    centroids[:, 1],
    marker="x",
    s=169,
    linewidths=3,
    color="w",
    zorder=10,
)
plt.title(
    "K-means clustering on the PCA-reduced data"
    ". Centroids are marked with white cross"
)

plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')

plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show
```

[24]: <function matplotlib.pyplot.show(close=None, block=None)>



0.8 EXPLAINING THE APPROACH

Step 1:Importing libraries Importing the neccessary libraries.

Step2:Data Exploration Creating dataframe from the given excel file. Replacing encypted pincode column with valid 6 digit unique numeric pincodes by creating a custom function. Using df.info() to check the count of number of rows,columns, non-null entries and features datatype.

Step3:Data Cleaning & Prepossesing using dataframe.isnull().sum() to check for the null values in dataframe. Plotting heatmap to visulize the positions of the null values in the dataframe. Since the sum of each rows is 100, imputing null values in agriculture with 0. Removing those rows where 'Multi Category' & 'Undefined' are None. Imputing the remaining Null values in the 'Undefined' and 'Multi Category' with the difference between 100% and the sum of the other known values in that row. checking if there any remaing null values. There are no null values remiaing. checking if the sum in each row is 100 Summary statistics of the cleaned dataframe. Checking for collinearity in categories features. The features appear to be highly collinear. Since there is an inherent mullticollinearity, using PCA to address it. Deriving the pricipal compnonents from the features.

Step4: Model Training Using Elbow method to determine Optimal clusters k, the optimal number of cluster are 6.

Step5: Training & Evaluating Clusstering Training and saving the model using Optimal k(derived by Elbow_point method). Evalauating the average silhouttee score of the clustering. The average score is 0.48 which is a decent score. Checking the dominant category in each cluster by derving the mean of each categorical feature in each cluster.

Step6 : Geographical Affinity Creating a new feature 'region' by extracting the first digit from the pincode. The first digit represent the region to which the pincode belongs to. Checking the

dominat regions in each cluster by identfying the region with maximum occurance in each cluster.