Code for Ford Ka_Group 26

September 24, 2023

0.0.1 Question 4

[1]:
```
import pandas as pd
import numpy as np
from scipy.stats import chi2_contingency
```

[2]:
```
df = pd.read_csv('data.csv')
```

[3]:
```
df.head()
```

[3]:

	Respondent Number	Preference Group	Gender	Age	Marital Status	\
0	1	1	2	44	3	
1	2	3	1	24	2	
2	3	2	2	34	3	
3	4	3	1	44	3	
4	5	1	2	41	1	

	Number of Children	1st Time Purchase	Age Category	Children Category	\
0	0	2	5	0	
1	1	1	1	1	
2	1	2	3	1	
3	0	2	5	0	
4	2	1	5	2	

	Income Category
0	6
1	3
2	1
3	3
4	4

[4]:
```
df = df.loc[df['Preference Group'] != 3]
```

[5]:
```
for column in df.columns.unique()[2:10]:
  tab = pd.crosstab(df[column],df['Preference Group'])
  chi2, p, dof, expected = chi2_contingency(tab.values)
  print("The p-value for preference group and group '%s' is: %.7f" % (column,p))
```

```
The p-value for preference group and group 'Gender' is: 0.7566210
The p-value for preference group and group 'Age' is: 0.6953644
The p-value for preference group and group 'Marital Status' is: 0.1675335
The p-value for preference group and group 'Number of Children' is: 0.3292833
The p-value for preference group and group '1st Time Purchase' is: 1.0000000
The p-value for preference group and group 'Age Category' is: 0.2397592
The p-value for preference group and group 'Children Category' is: 0.3561240
The p-value for preference group and group 'Income Category' is: 0.3785101
```

0.0.2 Question 5

[8]:
```
# import Python packages
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt

# regression package
import statsmodels.api as sm

# factor analysis packages
import factor_analyzer
# you could also use sklearn to run PCA

# clustering packages
from sklearn import cluster
from sklearn.cluster import KMeans
# k_means() is a wrapper that returns the result of KMeans.fit()
from yellowbrick.cluster import KElbowVisualizer
```

[9]:
```
survey = pd.read_csv('A1_Q5.csv')
# take a look at the bank data
survey.head(5)
```

[9]:

	Respondent Number	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	…	Q53	Q54	Q55	\
0	1	6	2	4	3	1	5	5	3	4	…	2	4	5	
1	2	7	7	7	5	4	4	5	4	5	…	1	1	1	
2	3	5	4	6	5	7	5	3	5	4	…	3	5	6	
3	4	4	2	5	4	2	4	5	4	3	…	3	5	4	
4	5	5	5	7	6	7	3	4	5	4	…	6	4	5	

	Q56	Q57	Q58	Q59	Q60	Q61	Q62
0	4	5	3	4	4	4	2
1	1	5	4	3	5	4	5
2	3	4	4	5	3	4	4
3	4	4	2	5	5	5	3
4	5	4	5	4	3	4	5

```
[5 rows x 63 columns]
```

[10]:
```
survey_X = survey.iloc[:, 1:]

survey_X_const = sm.add_constant(survey_X) ## use later for regression
survey_X.head(5)
```

[10]:

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	…	Q53	Q54	Q55	Q56	Q57	Q58	\
0	6	2	4	3	1	5	5	3	4	4	…	2	4	5	4	5	3	
1	7	7	7	5	4	4	5	4	5	5	…	1	1	1	1	5	4	
2	5	4	6	5	7	5	3	5	4	5	…	3	5	6	3	4	4	
3	4	2	5	4	2	4	5	4	3	4	…	3	5	4	4	4	2	
4	5	5	7	6	7	3	4	5	4	2	…	6	4	5	5	4	5	

	Q59	Q60	Q61	Q62
0	4	4	4	2
1	3	5	4	5
2	5	3	4	4
3	5	5	5	3
4	4	3	4	5

```
[5 rows x 62 columns]
```

[11]:
```
survey_pca = factor_analyzer.FactorAnalyzer(n_factors=10,
                                            rotation=None,
                                            method='principal').fit(survey_X)
```

[12]:
```
def get_loadings_communalities(pca,round_dig=2,index_names=None):
    df = pd.DataFrame(
      pca.loadings_,
      index=index_names if index_names else [f'q{i}' for i in range(1,1+pca.
 ↪loadings_.shape[0])],
      columns=[f'RC{i}' for i in range(1,1+pca.loadings_.shape[1])] if pca.
 ↪rotation else [f'PC{i}' for i in range(1,1+pca.loadings_.shape[1])]
      )
    if pca.rotation:
        df['communalities']=pca.get_communalities()
    df=df.round(3)
    return  df
get_loadings_communalities(survey_pca)
```

[12]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
q1	-0.545	-0.217	0.569	0.096	-0.005	0.017	0.022	-0.063	0.074	0.029
q2	-0.853	0.151	0.288	-0.016	0.023	0.046	0.006	-0.005	0.052	0.016
q3	0.175	0.660	-0.174	0.011	0.086	-0.009	0.058	0.039	-0.049	-0.137
q4	-0.061	0.617	-0.569	-0.088	-0.038	0.009	0.103	0.023	0.026	-0.034

```
q5  -0.174  0.799  0.261  0.033 -0.017  0.035  0.042 -0.007  0.001 -0.032
..     …      …      …      …      …      …      …      …      …      …
q58  0.234 -0.101  0.627 -0.038 -0.007 -0.086  0.102 -0.008 -0.085  0.026
q59  0.208 -0.210  0.609  0.001 -0.121  0.090  0.053  0.198  0.111  0.028
q60 -0.252  0.037 -0.648  0.077 -0.018  0.080 -0.068  0.098 -0.042 -0.035
q61 -0.401  0.130 -0.600  0.008  0.091 -0.026  0.111 -0.141  0.113  0.080
q62 -0.280  0.143 -0.545 -0.092  0.062  0.041 -0.013  0.024  0.033 -0.086

[62 rows x 10 columns]
```

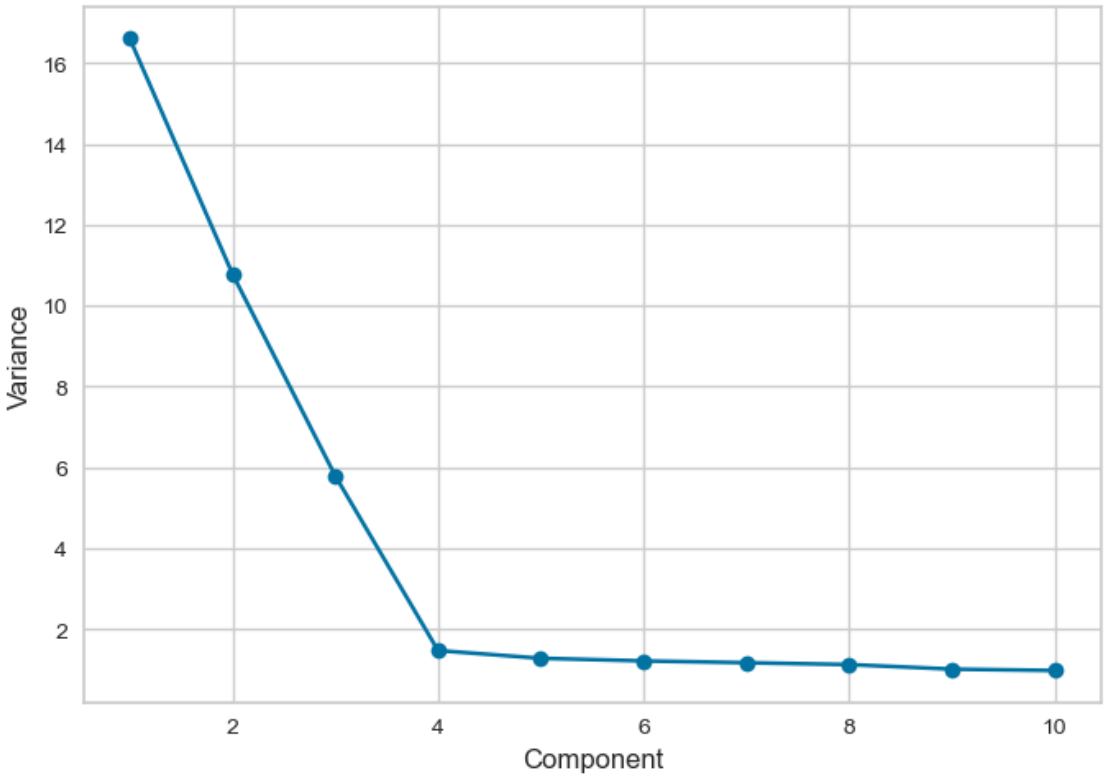
[13]:
```
## !!!!Don't run this code again! I already have the csv file!!!!!
loadings = get_loadings_communalities(survey_pca)
loadings.to_csv('loadings_output.csv', index=False)
```

[14]:
```
def get_summary(pca,round_dig=2):
  ''' Print a summary of the PCA fit '''
  return pd.DataFrame(
      [pca.get_factor_variance()[0],
       pca.get_factor_variance()[1],
       pca.get_factor_variance()[2]],
       columns=['PC{}'.format(i) for i in
                range(1,1+len(pca.get_factor_variance()[0]))],
      index=['Sum of Squares Loadings','Proportion of Variance Explained',
             'Cumulative Proportion']
            ).round(round_dig)

get_summary(survey_pca)
```

[14]:
```
                                   PC1    PC2   PC3   PC4   PC5   PC6   PC7  \
Sum of Squares Loadings          16.60  10.78  5.79  1.47  1.28  1.21  1.17
Proportion of Variance Explained  0.27   0.17  0.09  0.02  0.02  0.02  0.02
Cumulative Proportion             0.27   0.44  0.54  0.56  0.58  0.60  0.62

                                   PC8   PC9  PC10
Sum of Squares Loadings           1.12  1.01  0.98
Proportion of Variance Explained  0.02  0.02  0.02
Cumulative Proportion             0.64  0.65  0.67
```

[15]:
```
# scree plot - shows successive proportion of additional variance that each␣
 ↪component adds
import matplotlib.pyplot as plt
plt.plot(1+np.arange(len(survey_pca.get_factor_variance()[0])),
         survey_pca.get_factor_variance()[0],'o-')
plt.xlabel('Component')
plt.ylabel('Variance')
```

[15]:
```
Text(0, 0.5, 'Variance')
```



[16]:
```
## Rotation
survey_pca_rotated = factor_analyzer.FactorAnalyzer(n_factors=3,
                                                    rotation='varimax',
                                                    method='principal').fit(survey_X)
get_loadings_communalities(survey_pca_rotated)
```

[16]:

	RC1	RC2	RC3	communalities
q1	0.630	-0.322	0.409	0.668
q2	0.910	0.023	0.075	0.834
q3	-0.119	0.687	-0.101	0.497
q4	0.008	0.643	-0.544	0.708
q5	0.340	0.751	0.240	0.737
..	…	…	…	…
q58	-0.089	-0.115	0.661	0.458
q59	-0.084	-0.224	0.633	0.458
q60	0.093	0.050	-0.688	0.484
q61	0.260	0.120	-0.675	0.537
q62	0.159	0.144	-0.592	0.396

[62 rows x 4 columns]

[17]:
```
## !!!!Don't run this code again! I already have the csv file!!!!!
loadings_rotated = get_loadings_communalities(survey_pca_rotated)
loadings_rotated.to_csv('loadings_rotated_output.csv', index=True)
```

[18]:
```
##get scores
survey_X_scores = survey_pca_rotated.transform(survey_X)
survey_X_scores = pd.DataFrame(survey_X_scores,columns=['RC1','RC2','RC3'])
survey_X_scores.head(5)
## RC1
```

[18]:

	RC1	RC2	RC3
0	-1.002237	-1.042724	-0.706512
1	1.562683	-0.003871	-0.528081
2	-0.264483	1.857548	0.225793
3	-1.093728	-0.691215	-1.278540
4	-0.526620	1.718462	-0.033414

[19]:
```
get_summary(survey_pca_rotated)
## very similar
```

[19]:

	PC1	PC2	PC3
Sum of Squares Loadings	15.88	10.85	6.45
Proportion of Variance Explained	0.26	0.17	0.10
Cumulative Proportion	0.26	0.43	0.54

[20]:
```
def check_clusters(data,labels):
    print(list(zip(*np.unique(labels,return_counts=True))))
  # pivot_table() calculates the mean by default
    return pd.pivot_table(data,index=labels)
```

[21]:
```
## With 2 clusters
centroids_km, labels_km, inertia_km = cluster.
 ↪k_means(survey_X_scores,n_clusters=2,random_state=9650)
check_clusters(survey_X_scores,labels_km)
```

```
[(0, 143), (1, 107)]
```

```
/Users/deniseliang/anaconda3/lib/python3.10/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
```

[21]:

	RC1	RC2	RC3
0	0.663263	0.511923	-0.159378
1	-0.886416	-0.684159	0.213001

[22]:
```
##With 3 clusters
centroids_km, labels_km, inertia_km = cluster.
 ↪k_means(survey_X_scores,n_clusters=3,random_state=9650)
check_clusters(survey_X_scores,labels_km)
```

```
[(0, 78), (1, 107), (2, 65)]
```

```
/Users/deniseliang/anaconda3/lib/python3.10/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
```

[22]:

	RC1	RC2	RC3
0	1.383266	-0.430916	-0.245159
1	-0.886416	-0.684159	0.213001
2	-0.200742	1.643329	-0.056442

[23]:
```
## With 4 clusters
centroids_km, labels_km, inertia_km = cluster.
 ↪k_means(survey_X_scores,n_clusters=4,random_state=9650)
check_clusters(survey_X_scores,labels_km)
```

```
[(0, 78), (1, 75), (2, 32), (3, 65)]
```

```
/Users/deniseliang/anaconda3/lib/python3.10/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
```

[23]:

	RC1	RC2	RC3
0	1.383266	-0.430916	-0.245159
1	-1.066228	-0.748296	-0.722522
2	-0.464984	-0.533837	2.405634
3	-0.200742	1.643329	-0.056442

[24]:
```
## With 5 clusters
centroids_km, labels_km, inertia_km = cluster.
 ↪k_means(survey_X_scores,n_clusters=5,random_state=9650)
check_clusters(survey_X_scores,labels_km)
```

```
[(0, 78), (1, 65), (2, 37), (3, 32), (4, 38)]
```

```
/Users/deniseliang/anaconda3/lib/python3.10/site-
packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  warnings.warn(
```

[24]:

	RC1	RC2	RC3
0	1.383266	-0.430916	-0.245159
1	-0.200742	1.643329	-0.056442
2	-1.024770	-0.821611	-0.481276
3	-0.464984	-0.533837	2.405634
4	-1.106594	-0.676910	-0.957421