Project2

September 23, 2022

[5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

data = pd.read_csv('mcdonalds.csv')
data.head()
```

[5]:

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	disgusting	Like	Age	VisitFrequency	Gender
0	No	Yes	No	Yes	No	Yes	Yes	No	Yes	No	No	-3	61	Every three months	Female
1	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No	+2	51	Every three months	Female
2	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	+1	62	Every three months	Female
3	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	No	Yes	+4	69	Once a week	Female
4	No	Yes	No	Yes	Yes	Yes	Yes	No	No	Yes	No	+2	49	Once a month	Male

[6]:
```
data.isna().sum()
```

[6]:
```
yummy           0
convenient      0
spicy           0
fattening       0
greasy          0
fast            0
cheap           0
tasty           0
expensive       0
healthy         0
disgusting      0
Like            0
Age             0
```

```
VisitFrequency    0
Gender            0
dtype: int64
```

[7]:
```
from sklearn.preprocessing import LabelEncoder
def label(x):
 data[x] = LabelEncoder().fit_transform(data[x])
 return data

cat = ['yummy', 'convenient', 'spicy', 'fattening', 'greasy', 'fast', 'cheap',
 'tasty', 'expensive', 'healthy', 'disgusting']

for i in cat:
 label(i)
```

[8]:
```
data
```

[8]:

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty
0	0	1	0	1	0	1	1	0
1	1	1	0	1	1	1	1	1
2	0	1	1	1	1	1	0	1
3	1	1	0	1	1	1	1	1
4	0	1	0	1	1	1	1	0
…	…	…	…	…	…	…	…	…
1448	0	1	0	1	1	0	0	0
1449	1	1	0	1	0	0	1	1
1450	1	1	0	1	0	1	0	1
1451	1	1	0	0	0	1	1	1
1452	0	1	0	1	1	0	0	0

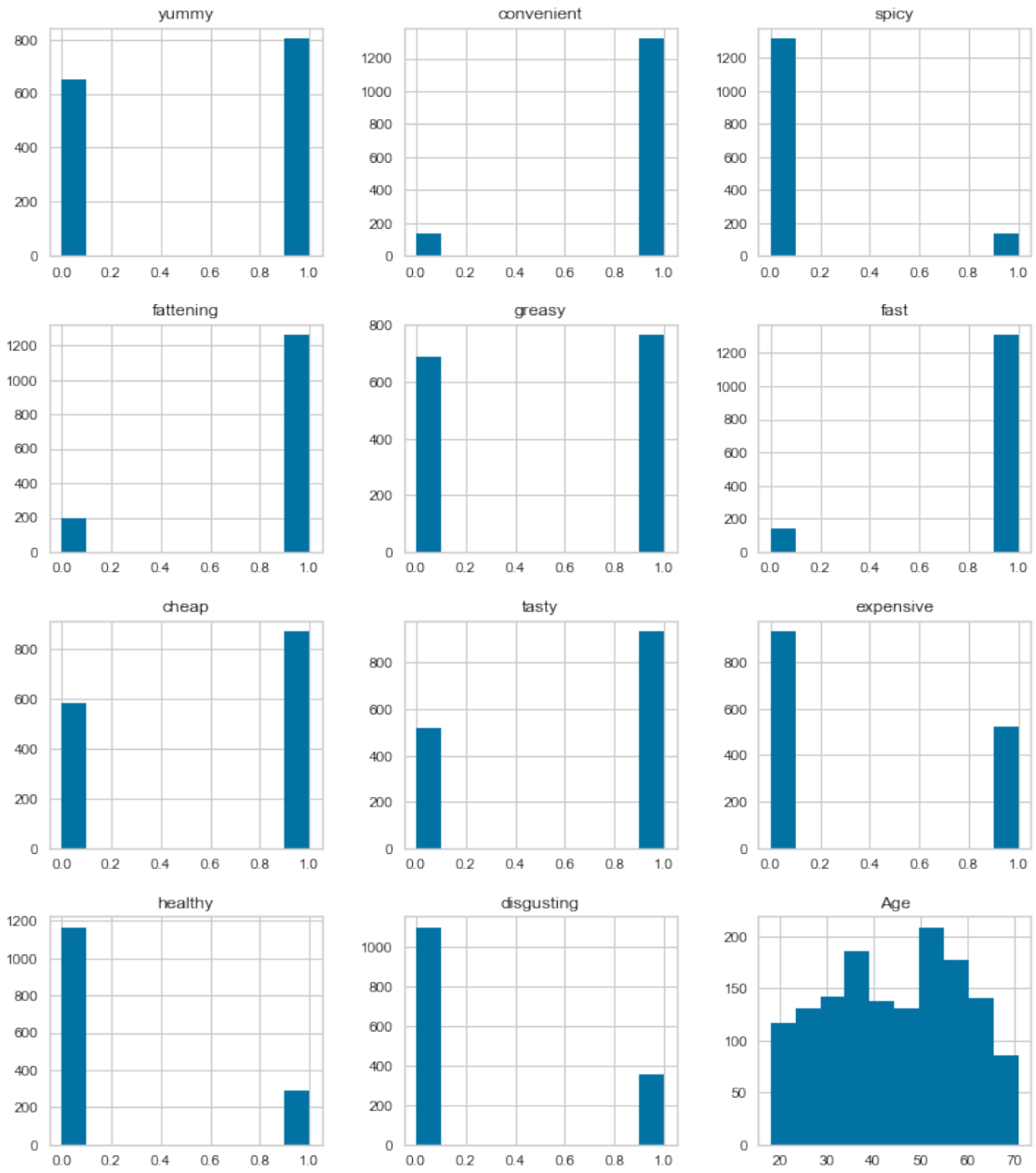
	expensive	healthy	disgusting	Like	Age	VisitFrequency
0	1	0	0	-3	61	Every three months
1	1	0	0	+2	51	Every three months
2	1	1	0	+1	62	Every three months
3	0	0	1	+4	69	Once a week
4	0	1	0	+2	49	Once a month
…	…	…	…	…	…	…
1448	1	0	1	I hate it!-5	47	Once a year
1449	0	1	0	+2	36	Once a week
1450	1	0	0	+3	52	Once a month
1451	0	1	0	+4	41	Every three months
1452	1	0	1	-3	30	Every three months

	Gender
0	Female
1	Female
2	Female

```
3     Female
4       Male
…       …
1448    Male
1449  Female
1450  Female
1451    Male
1452    Male

[1453 rows x 15 columns]
```

[9]:
```
plt.rcParams['figure.figsize'] = (12,14)
data.hist()
plt.show()
```



[10]:
```
data_one = data.loc[:,cat]
data_one
```

[10]:

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	\
0	0	1	0	1	0	1	1	0	
1	1	1	0	1	1	1	1	1	
2	0	1	1	1	1	1	0	1	
3	1	1	0	1	1	1	1	1	
4	0	1	0	1	1	1	1	0	

```
…     …          …    …         …     …   …    …    …
1448      0           1      0          1       1     0      0      0
1449      1           1      0          1       0     0      1      1
1450      1           1      0          1       0     1      0      1
1451      1           1      0          0       0     1      1      1
1452      0           1      0          1       1     0      0      0

      expensive  healthy  disgusting
0             1        0           0
1             1        0           0
2             1        1           0
3             0        0           1
4             0        1           0
…           …      …          …
1448          1        0           1
1449          0        1           0
1450          1        0           0
1451          0        1           0
1452          1        0           1

[1453 rows x 11 columns]
```

[11]:
```
from sklearn.decomposition import PCA
from sklearn import preprocessing
x = data.loc[:,cat].values
pca_data = preprocessing.scale(x)
pca = PCA(n_components=11)
pc = pca.fit_transform(x)
names = ['pc1','pc2','pc3','pc4','pc5','pc6','pc7','pc8','pc9','pc10','pc11']
pf = pd.DataFrame(data = pc, columns = names)
pf.head()
```

[11]:
```
        pc1       pc2       pc3       pc4       pc5       pc6       pc7  \
0  0.425367 -0.219079  0.663255 -0.401300  0.201705 -0.389767 -0.211982
1 -0.218638  0.388190 -0.730827 -0.094724  0.044669 -0.086596 -0.095877
2  0.375415  0.730435 -0.122040  0.692262  0.839643 -0.687406  0.583112
3 -0.172926 -0.352752 -0.843795  0.206998 -0.681415 -0.036133 -0.054284
4  0.187057 -0.807610  0.028537  0.548332  0.854074 -0.097305 -0.457043

        pc8       pc9      pc10      pc11
0  0.163235  0.181007  0.515706 -0.567074
1 -0.034756  0.111476  0.493313 -0.500440
2  0.364379 -0.322288  0.061759  0.242741
3 -0.231477 -0.028003 -0.250678 -0.051034
4  0.171758 -0.074409  0.031897  0.082245
```

[12]:
```
loadings = pca.components_
num_pc = pca.n_features_
pc_list = ["PC"+str(i) for i in list(range(1, num_pc+1))]
loadings_df = pd.DataFrame.from_dict(dict(zip(pc_list, loadings)))
loadings_df['variable'] = data_one.columns.values
loadings_df = loadings_df.set_index('variable')
loadings_df
```

[12]:

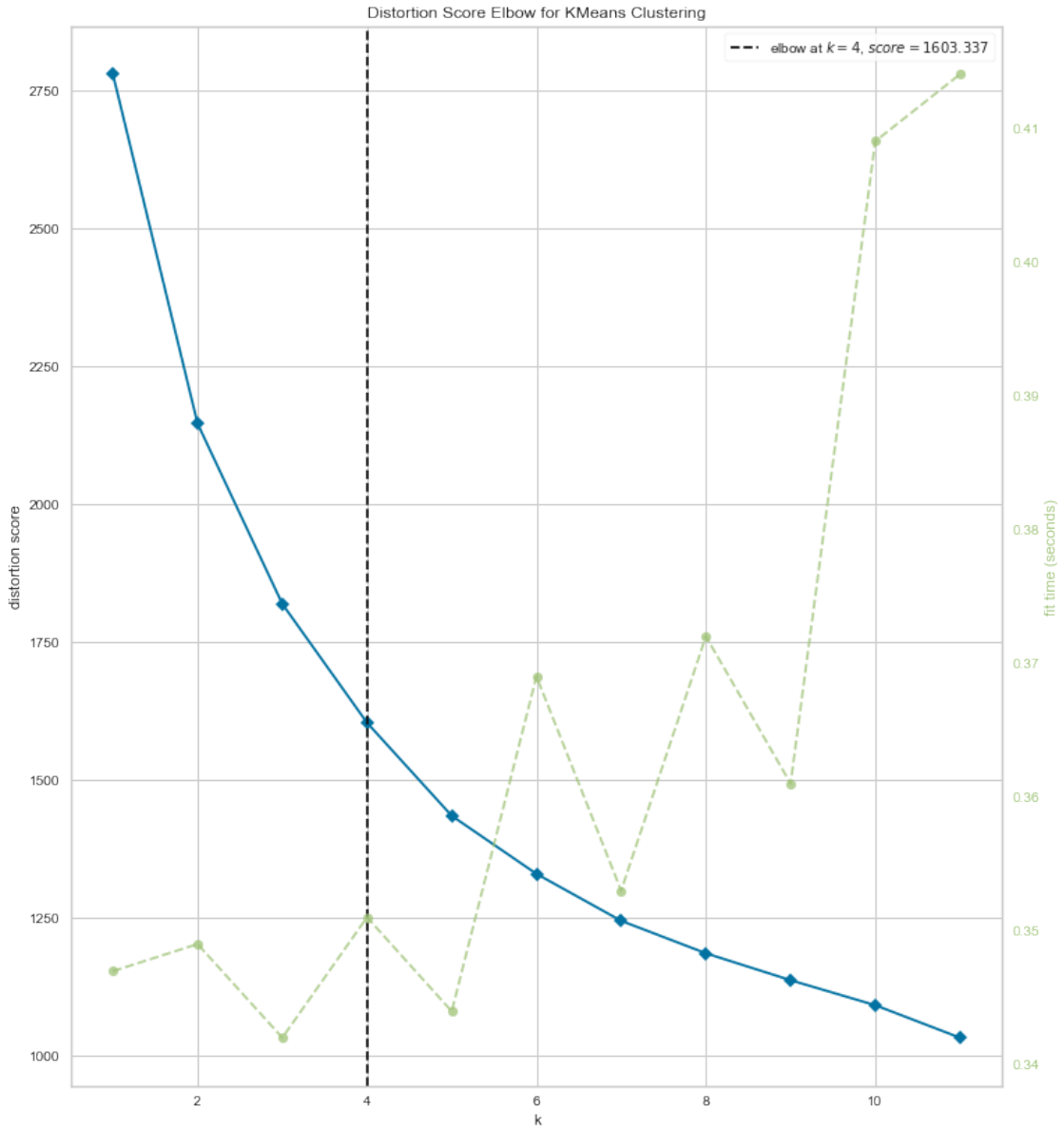
variable	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11
yummy	-0.476933	0.363790	-0.304444	0.055162	-0.307535	0.170738	-0.280519	0.013041	0.572403	-0.110284	0.045439
convenient	-0.155332	0.016414	-0.062515	-0.142425	0.277608	-0.347830	-0.059738	-0.113079	-0.018465	-0.665818	-0.541616
spicy	-0.006356	0.018809	-0.037019	0.197619	0.070620	-0.355087	0.707637	0.375934	0.400280	-0.075634	0.141730
fattening	0.116232	-0.034094	-0.322359	-0.354139	-0.073405	-0.406515	-0.385943	0.589622	-0.160512	-0.005338	0.250910
greasy	0.304443	-0.063839	-0.802373	0.253960	0.361399	0.209347	0.036170	-0.138241	-0.002847	0.008707	0.001642
fast	-0.108493	-0.086972	-0.064642	-0.097363	0.107930	-0.594632	-0.086846	-0.627799	0.166197	0.239532	0.339265
cheap	-0.337186	-0.610633	-0.149310	0.118958	-0.128973	-0.103241	-0.040449	0.140060	0.076069	0.428087	-0.489283
tasty	-0.471514	0.307318	-0.287265	-0.002547	-0.210899	-0.076914	0.360453	-0.072792	-0.639086	0.079184	0.019552
expensive	0.329042	0.601286	0.024397	0.067816	-0.003125	-0.261342	-0.068385	0.029539	0.066996	0.454399	-0.490069
healthy	-0.213711	0.076593	0.192051	0.763488	0.287846	-0.178226	-0.349616	0.176303	-0.185572	-0.038117	0.157608
disgusting	0.374753	-0.139656	-0.088571	0.369539	-0.729209	-0.210878	-0.026792	-0.167181	-0.072483	-0.289592	-0.040662

[15]:
```
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,12)).fit(data_one)
visualizer.show()
```

```
C:\Users\ojasva\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=6.
  warnings.warn(
C:\Users\ojasva\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
```

```
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=6.
  warnings.warn(
C:\Users\ojasva\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=6.
  warnings.warn(
C:\Users\ojasva\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=6.
  warnings.warn(
C:\Users\ojasva\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=6.
  warnings.warn(
C:\Users\ojasva\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=6.
  warnings.warn(
C:\Users\ojasva\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=6.
  warnings.warn(
C:\Users\ojasva\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=6.
  warnings.warn(
C:\Users\ojasva\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=6.
  warnings.warn(
C:\Users\ojasva\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=6.
  warnings.warn(
C:\Users\ojasva\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1334:
UserWarning: KMeans is known to have a memory leak on Windows with MKL, when
there are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=6.
```

```
warnings.warn(
```



[15]:
```
<AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'},
xlabel='k', ylabel='distortion score'>
```

[ ]: