k-means

August 27, 2023

```
[1]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.cluster import KMeans
     from sklearn.preprocessing import LabelEncoder
```

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

```
[3]: data
```

```
[3]:      CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
     0             1    Male   19                  15                      39
     1             2    Male   21                  15                      81
     2             3  Female   20                  16                       6
     3             4  Female   23                  16                      77
     4             5  Female   31                  17                      40
     ..          …       …   …                 …                      …
     195         196  Female   35                 120                      79
     196         197  Female   45                 126                      28
     197         198    Male   32                 126                      74
     198         199    Male   32                 137                      18
     199         200    Male   30                 137                      83

     [200 rows x 5 columns]
```

```
[4]: # Data Info :
```

```
[5]: data.shape
```

```
[5]: (200, 5)
```

```
[6]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
```

```
---  ------               --------------  -----
 0   CustomerID           200 non-null    int64
 1   Gender               200 non-null    object
 2   Age                  200 non-null    int64
 3   Annual Income (k$)   200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

```
[7]: data.describe()
```

```
[7]:        CustomerID         Age  Annual Income (k$)  Spending Score (1-100)
     count  200.000000  200.000000          200.000000              200.000000
     mean   100.500000   38.850000           60.560000               50.200000
     std     57.879185   13.969007           26.264721               25.823522
     min      1.000000   18.000000           15.000000                1.000000
     25%     50.750000   28.750000           41.500000               34.750000
     50%    100.500000   36.000000           61.500000               50.000000
     75%    150.250000   49.000000           78.000000               73.000000
     max    200.000000   70.000000          137.000000               99.000000
```

```
[8]: data.drop(['CustomerID'],axis=1,inplace=True)           #Delete CustomerID from␣
       ↪data
```

```
[9]: data
```

```
[9]:      Gender  Age  Annual Income (k$)  Spending Score (1-100)
     0      Male   19                  15                      39
     1      Male   21                  15                      81
     2    Female   20                  16                       6
     3    Female   23                  16                      77
     4    Female   31                  17                      40
     ..      ...  ...                 ...                     ...
     195  Female   35                 120                      79
     196  Female   45                 126                      28
     197    Male   32                 126                      74
     198    Male   32                 137                      18
     199    Male   30                 137                      83

     [200 rows x 4 columns]
```

```
[10]: la=LabelEncoder()          #to convert object column to numerical
```

```
[11]: data['Gender']=la.fit_transform(data['Gender'])
```

```
[12]: data
```

[12]:
```
     Gender  Age  Annual Income (k$)  Spending Score (1-100)
0         1   19                  15                      39
1         1   21                  15                      81
2         0   20                  16                       6
3         0   23                  16                      77
4         0   31                  17                      40
..      …   …                 …                     …
195       0   35                 120                      79
196       0   45                 126                      28
197       1   32                 126                      74
198       1   32                 137                      18
199       1   30                 137                      83

[200 rows x 4 columns]
```

[13]:
```
#Apply the KMeans
```

[14]:
```
no_clusters=[]     # to store the values of clusters
j=[]
```

[15]:
```
for i in range(1,10):
    model=KMeans(n_clusters=i)
    model.fit(data)
    no_clusters.append(i)
    j.append(model.inertia_)
```

```
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-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(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
```

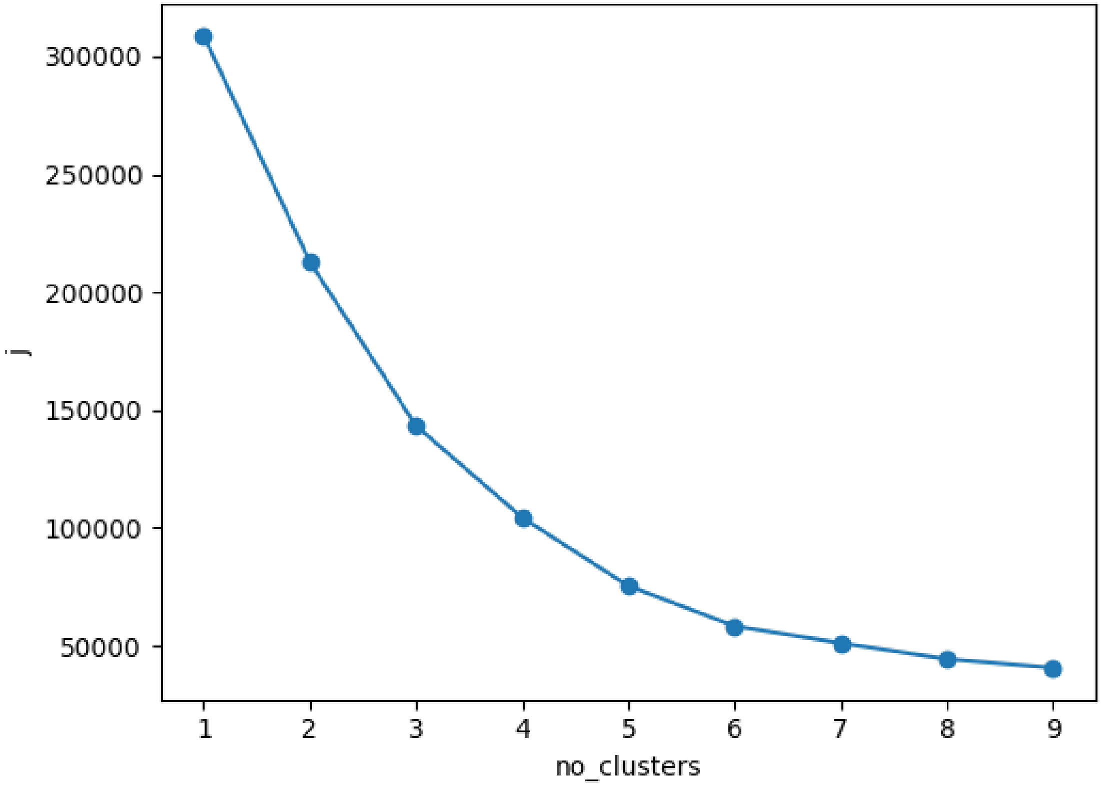
```
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10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

[16]:
```
pd.DataFrame(no_clusters,j)      #make DataFrame contain (no_clusters,j)
```

[16]:
```
                 0
308862.060000   1
212889.442455   2
143391.592360   3
104414.675342   4
75399.615414    5
58348.641363    6
51165.184237    7
44391.820805    8
40639.660395    9
```

[17]:
```
plt.plot(no_clusters,j,marker='o')
plt.xlabel('no_clusters')
plt.ylabel('j')
```

[17]:
```
Text(0, 0.5, 'j')
```



```
[19]: model=KMeans(n_clusters=5)
      model.fit(data)
      pre=model.predict(data)
```

```
/usr/local/lib/python3.10/dist-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(
```

```
[20]: data['KMeans']=pre      #add KMeans column to data
```

```
[21]: data
```

```
[21]:      Gender  Age  Annual Income (k$)  Spending Score (1-100)  KMeans
      0         1   19                  15                      39       1
      1         1   21                  15                      81       4
      2         0   20                  16                       6       1
      3         0   23                  16                      77       4
      4         0   31                  17                      40       1
      ..      …   …                 …                      …      …
      195       0   35                 120                      79       0
      196       0   45                 126                      28       2
```

```
197      1   32               126                  74      0
198      1   32               137                  18      2
199      1   30               137                  83      0

[200 rows x 5 columns]
```

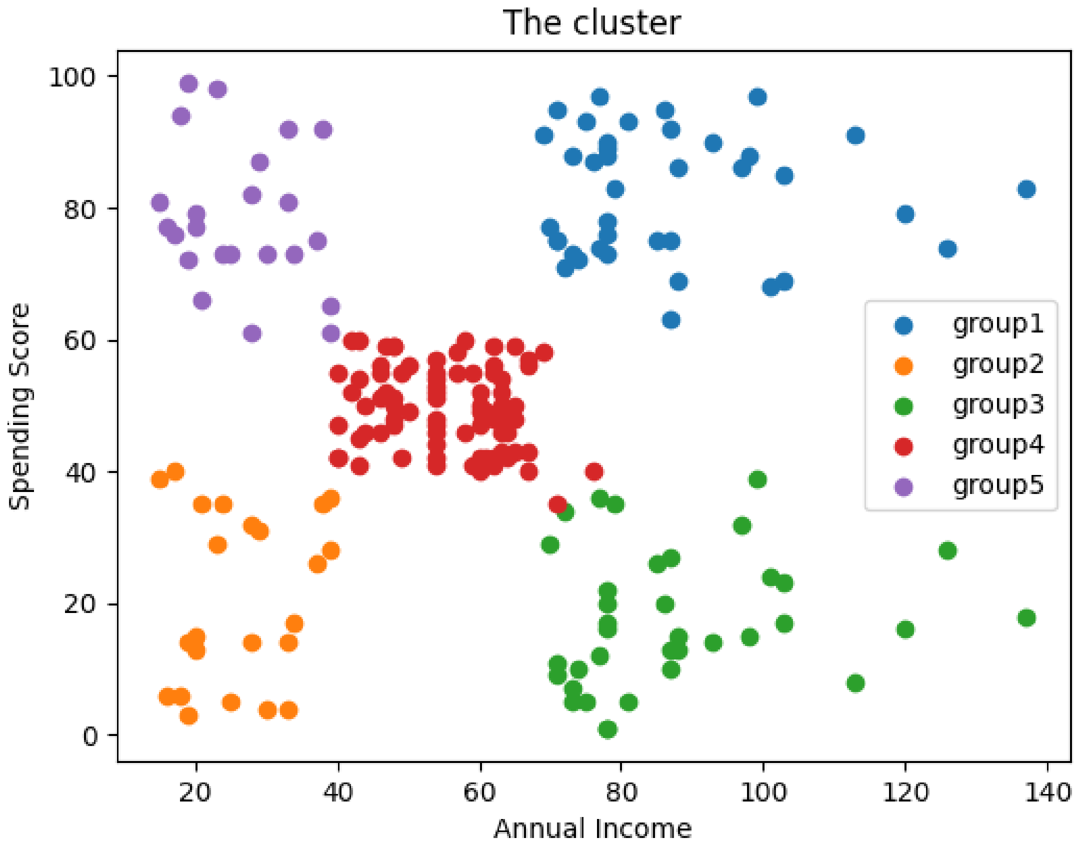
[22]:
```
group1=data[data['KMeans']==0]
group2=data[data['KMeans']==1]
group3=data[data['KMeans']==2]
group4=data[data['KMeans']==3]
group5=data[data['KMeans']==4]
```

[23]:
```
# Numerical Features vs Numerical Features w.r.t Categorical Feature
```

[24]:
```
#plot the final cluster
```

[25]:
```
plt.scatter(group1['Annual Income (k$)'],group1['Spending Score␣
  ↪(1-100)'],label='group1')
plt.scatter(group2['Annual Income (k$)'],group2['Spending Score␣
  ↪(1-100)'],label='group2')
plt.scatter(group3['Annual Income (k$)'],group3['Spending Score␣
  ↪(1-100)'],label='group3')
plt.scatter(group4['Annual Income (k$)'],group4['Spending Score␣
  ↪(1-100)'],label='group4')
plt.scatter(group5['Annual Income (k$)'],group5['Spending Score␣
  ↪(1-100)'],label='group5')
plt.legend()
plt.title('The cluster')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
```

[25]:
```
Text(0, 0.5, 'Spending Score')
```



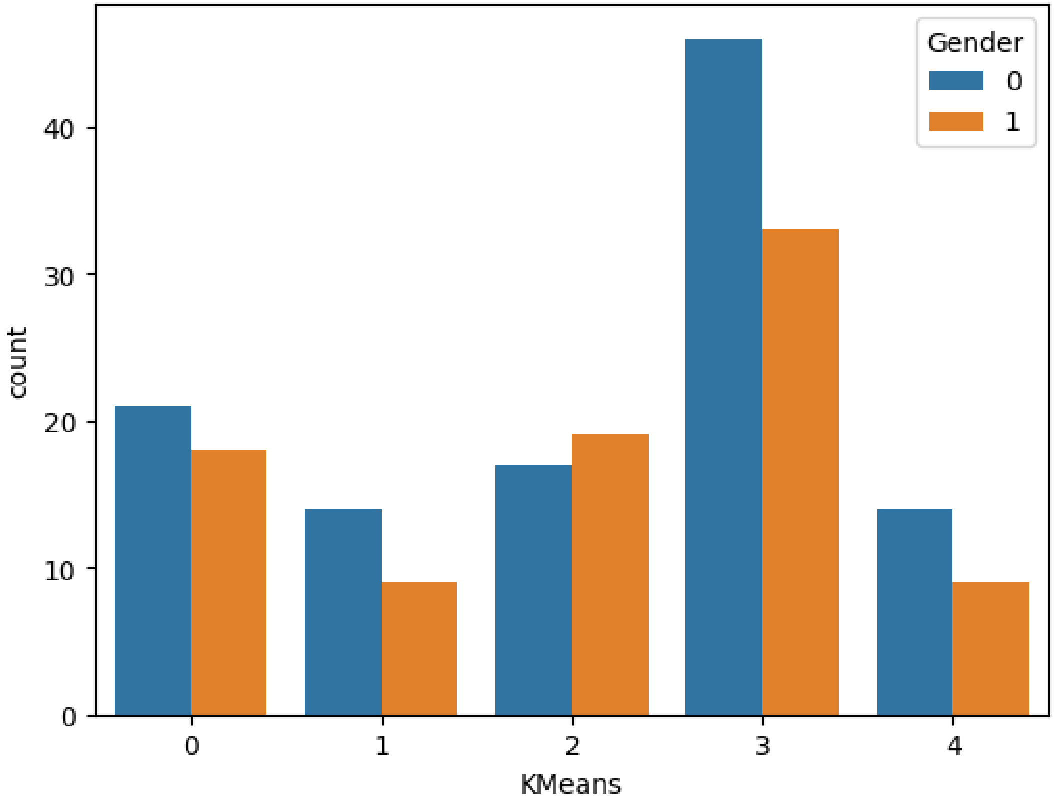
[26]:
```
data['KMeans'].value_counts()
```

[26]:
```
3    79
0    39
2    36
1    23
4    23
Name: KMeans, dtype: int64
```

[27]:
```
##Distribution graphs
```

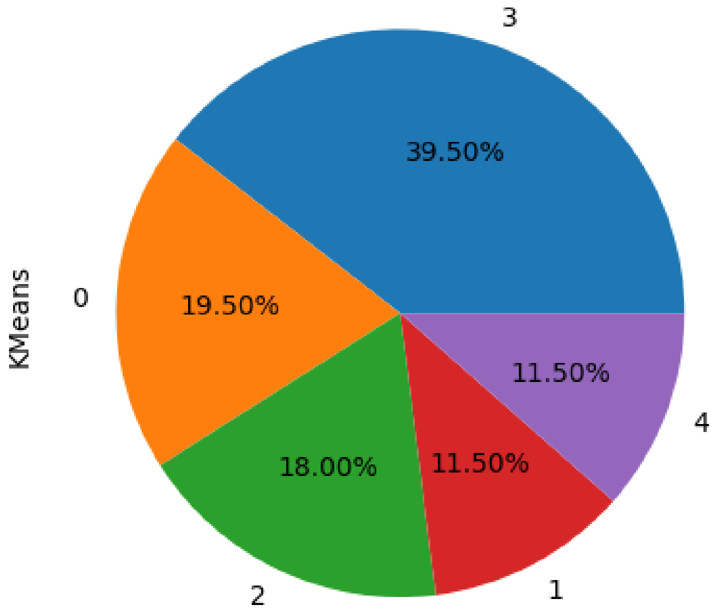
[28]:
```
sns.countplot(data,x='KMeans',hue='Gender')
```

[28]:
```
<Axes: xlabel='KMeans', ylabel='count'>
```



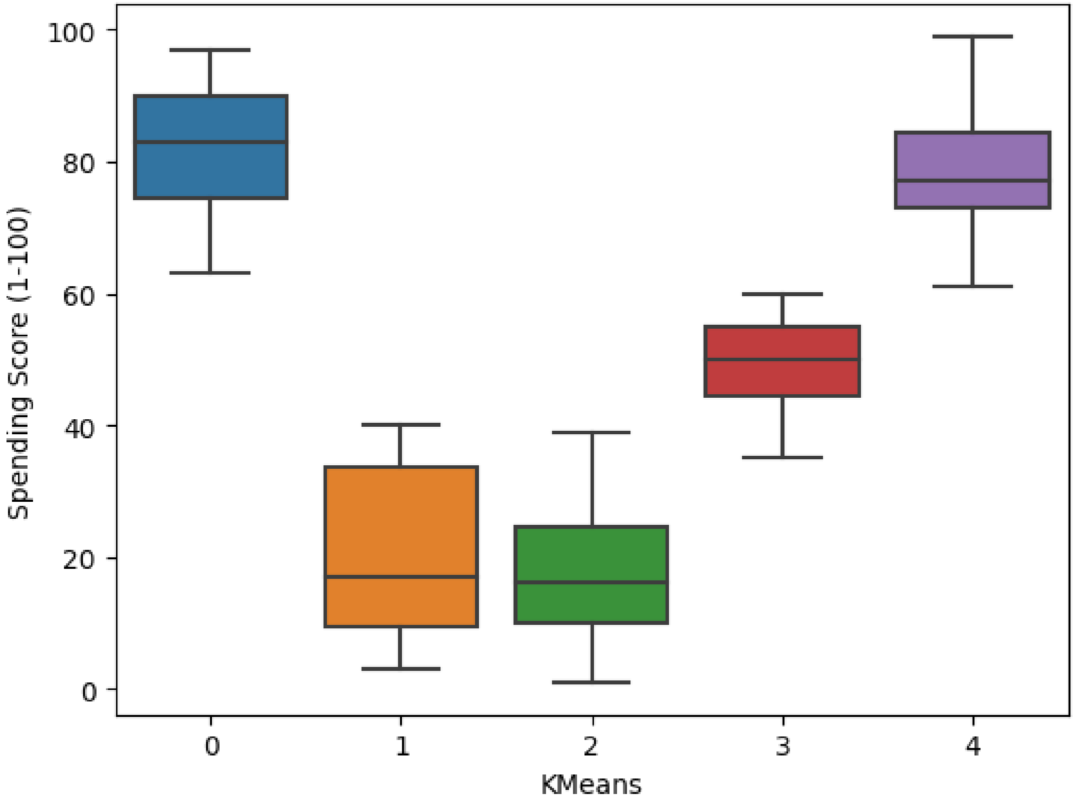
```
[29]: data['KMeans'].value_counts().plot.pie(autopct='%0.2f%%')
```

```
[29]: <Axes: ylabel='KMeans'>
```



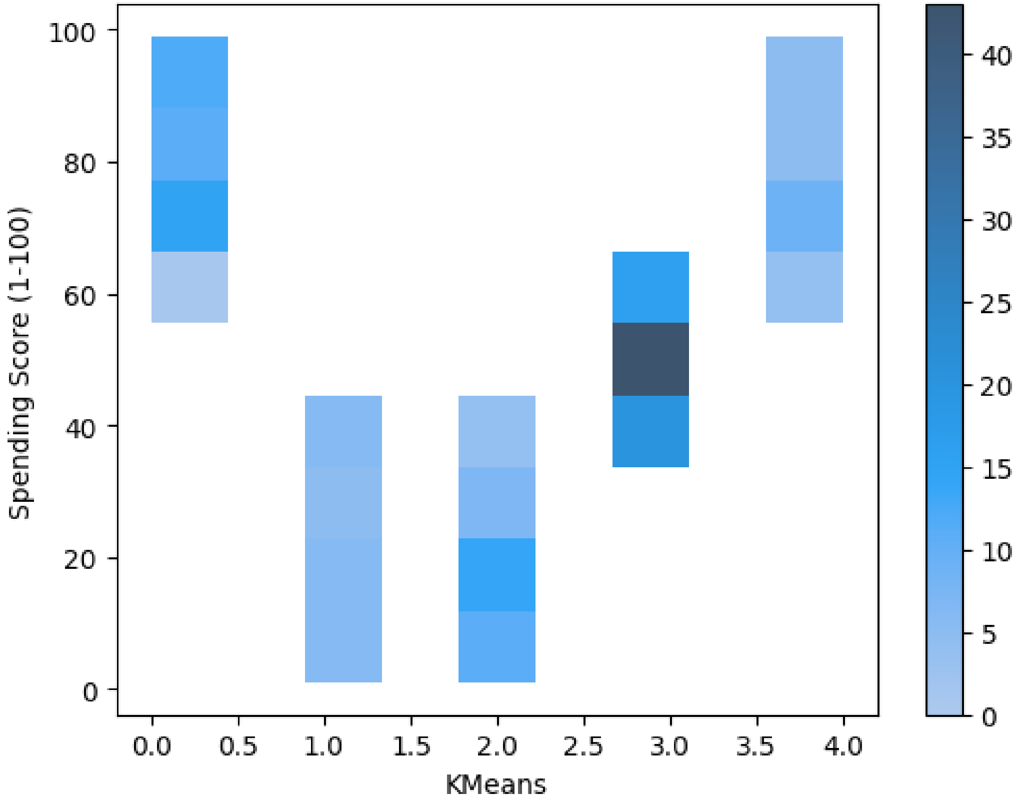
```
[30]: sns.boxplot(data=data,x='KMeans',y='Spending Score (1-100)')
```

```
[30]: <Axes: xlabel='KMeans', ylabel='Spending Score (1-100)'>
```



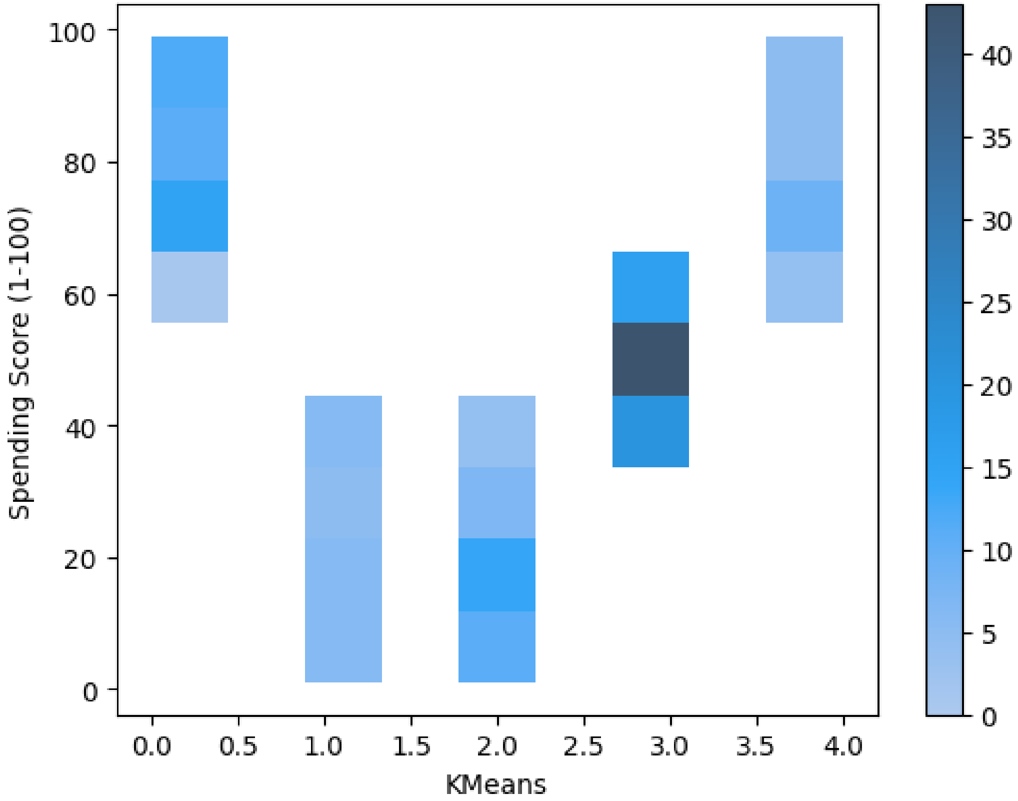
```
[32]: sns.histplot(data,x='KMeans',y='Spending Score (1-100)',cbar=True)
```

```
[32]: <Axes: xlabel='KMeans', ylabel='Spending Score (1-100)'>
```



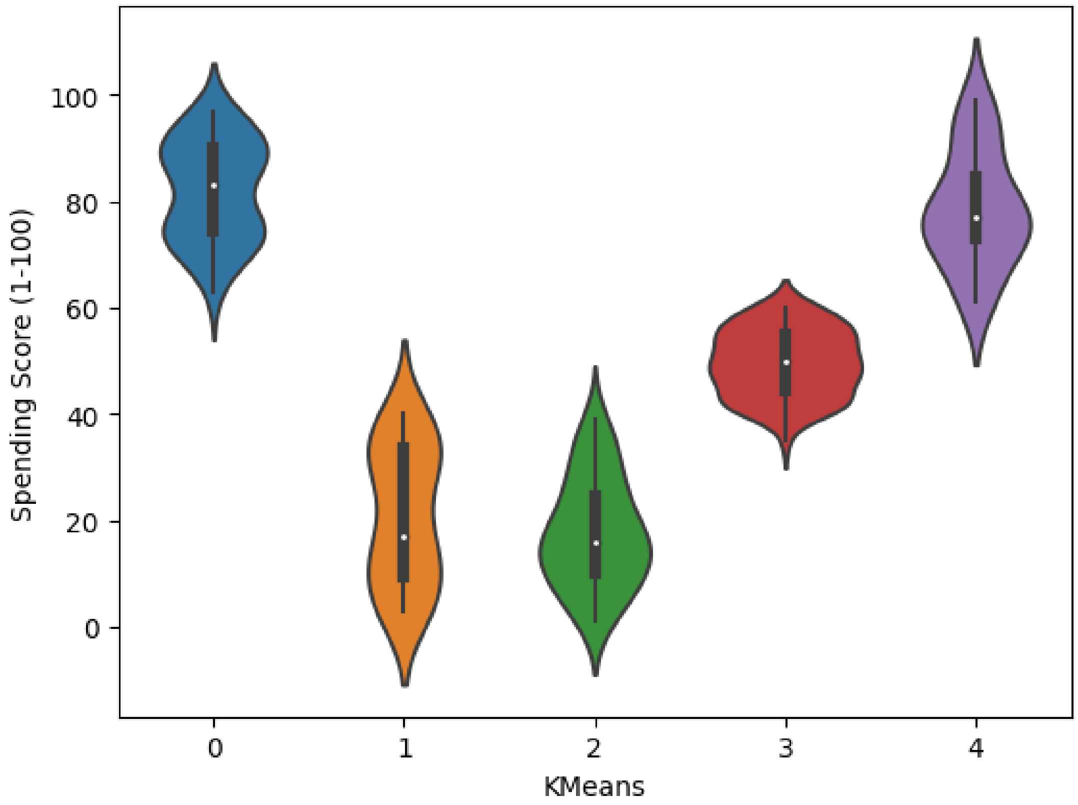
```
[33]: sns.histplot(data,x='KMeans',y='Spending Score (1-100)',cbar=True)
```

```
[33]: <Axes: xlabel='KMeans', ylabel='Spending Score (1-100)'>
```



```
[34]: sns.violinplot(data=data, x="KMeans", y="Spending Score (1-100)")  #to note the␣
      ↪density
```

```
[34]: <Axes: xlabel='KMeans', ylabel='Spending Score (1-100)'>
```



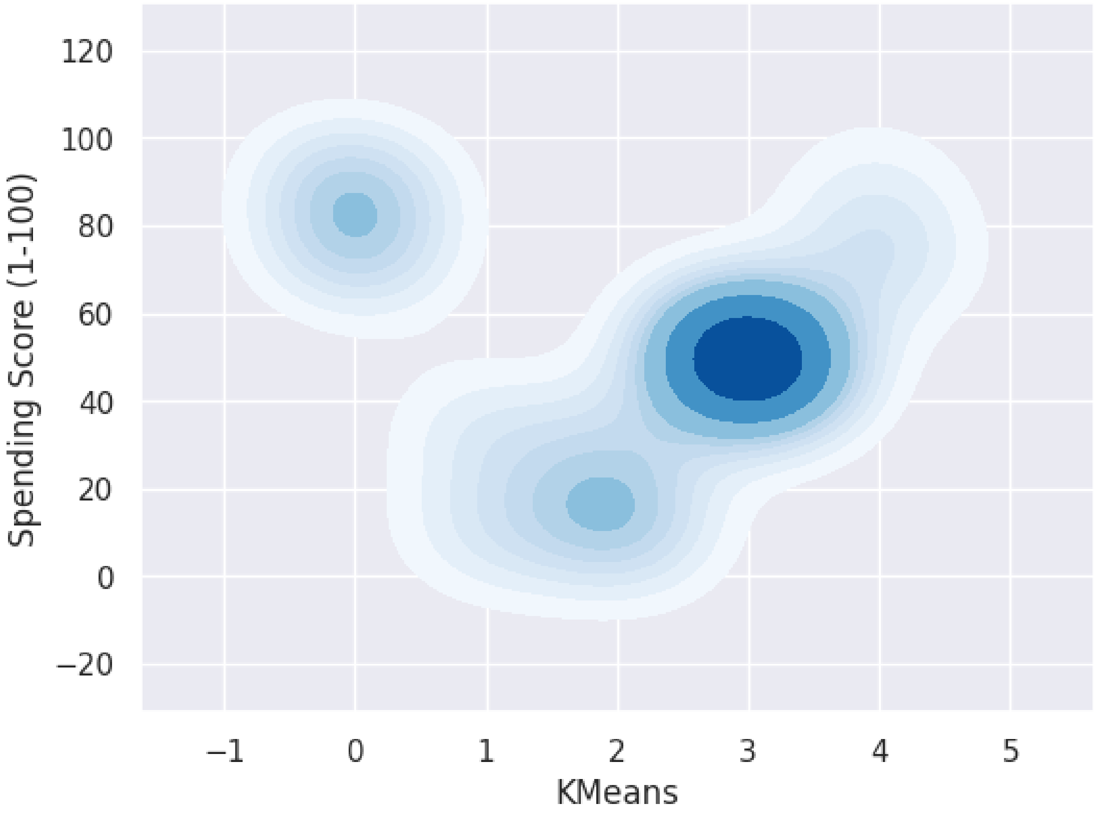
[39]:

```
sns.set_theme(style="darkgrid")

sns.kdeplot(data=data,x='KMeans',y='Spending Score (1-100)',thresh=.
 ↪1,cmap='Blues',shade=True)
```

```
<ipython-input-39-1cdbe89fb08e>:3: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data=data,x='KMeans',y='Spending Score
(1-100)',thresh=.1,cmap='Blues',shade=True)
```

[39]:

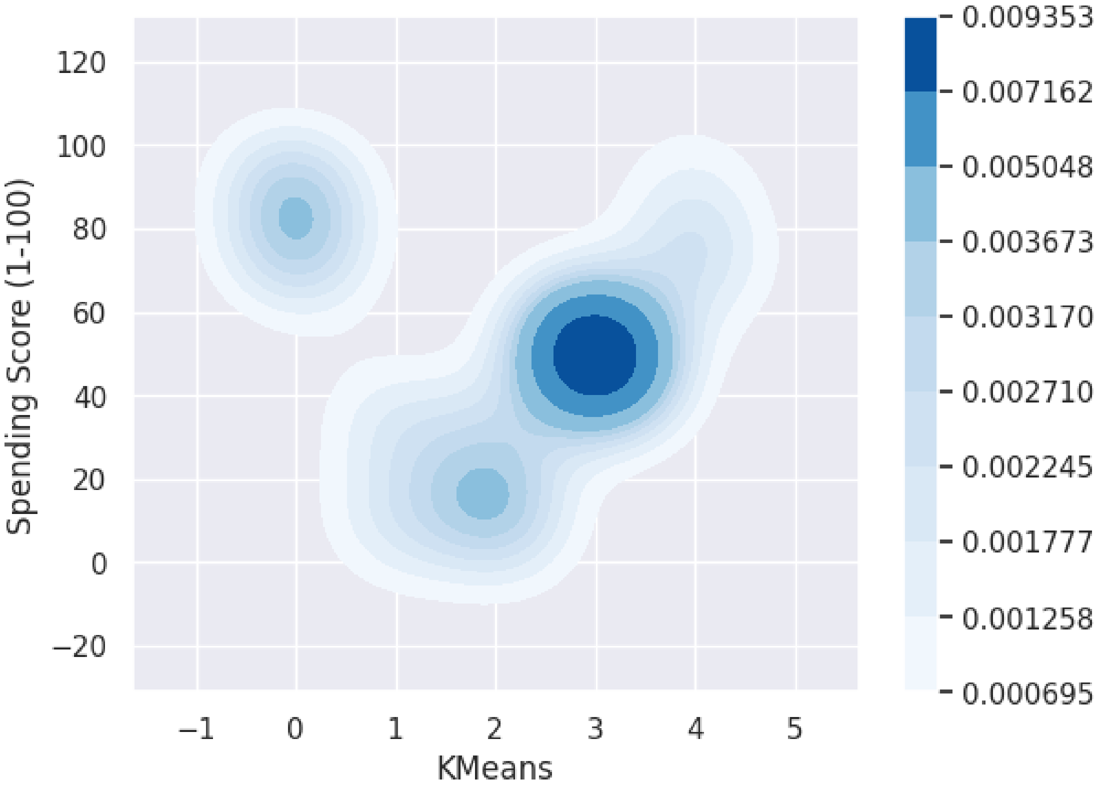
```
<Axes: xlabel='KMeans', ylabel='Spending Score (1-100)'>
```



[37]:
```
sns.set_theme(style="darkgrid")

sns.kdeplot(data=data,x='KMeans',y='Spending Score (1-100)',thresh=.
↪1,cmap='Blues',shade=True,cbar=True)
```

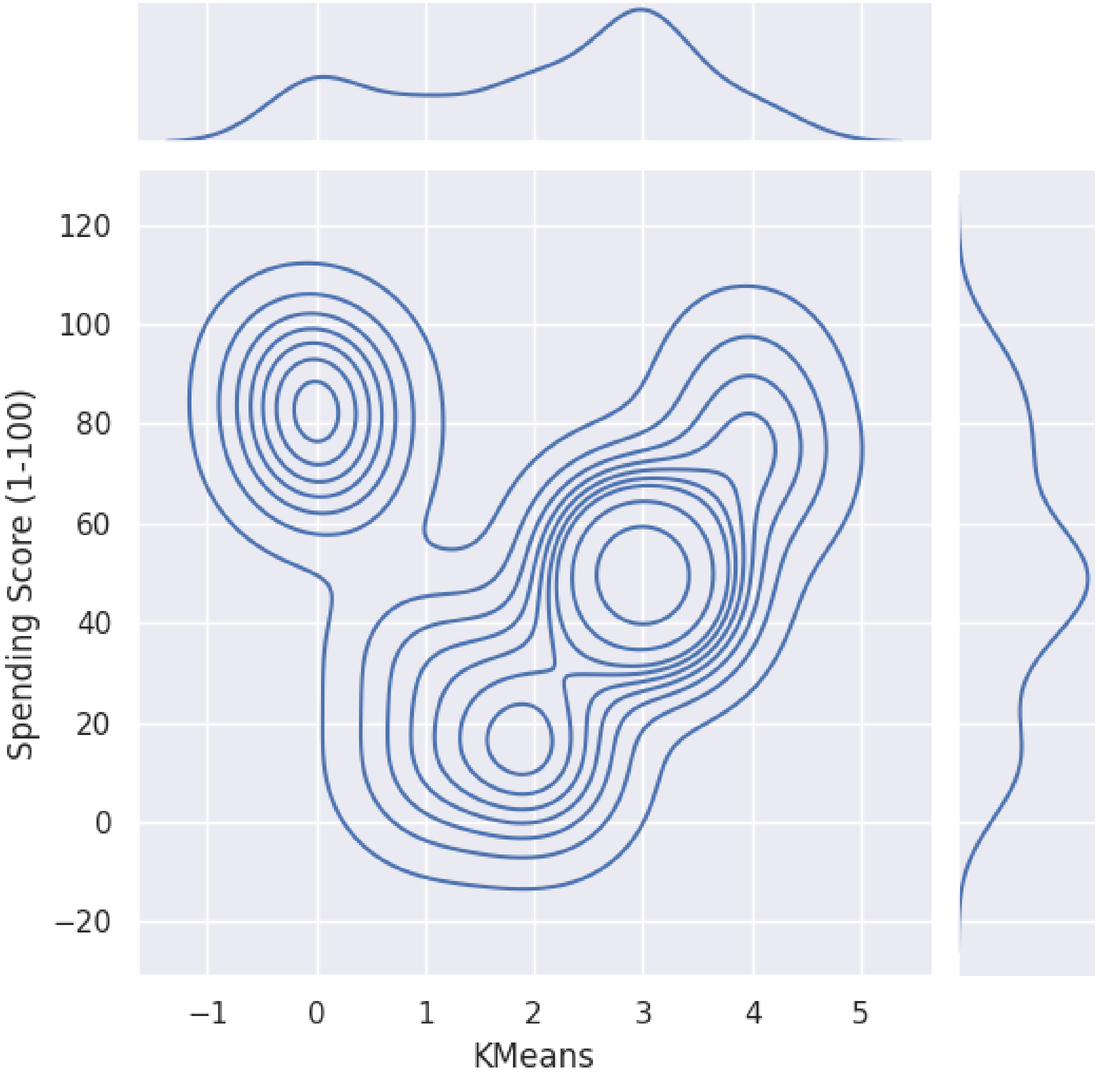
```
<ipython-input-37-f3668bc6c981>:3: FutureWarning:

`shade` is now deprecated in favor of `fill`; setting `fill=True`.
This will become an error in seaborn v0.14.0; please update your code.

  sns.kdeplot(data=data,x='KMeans',y='Spending Score
(1-100)',thresh=.1,cmap='Blues',shade=True,cbar=True)
```

[37]:
```
<Axes: xlabel='KMeans', ylabel='Spending Score (1-100)'>
```



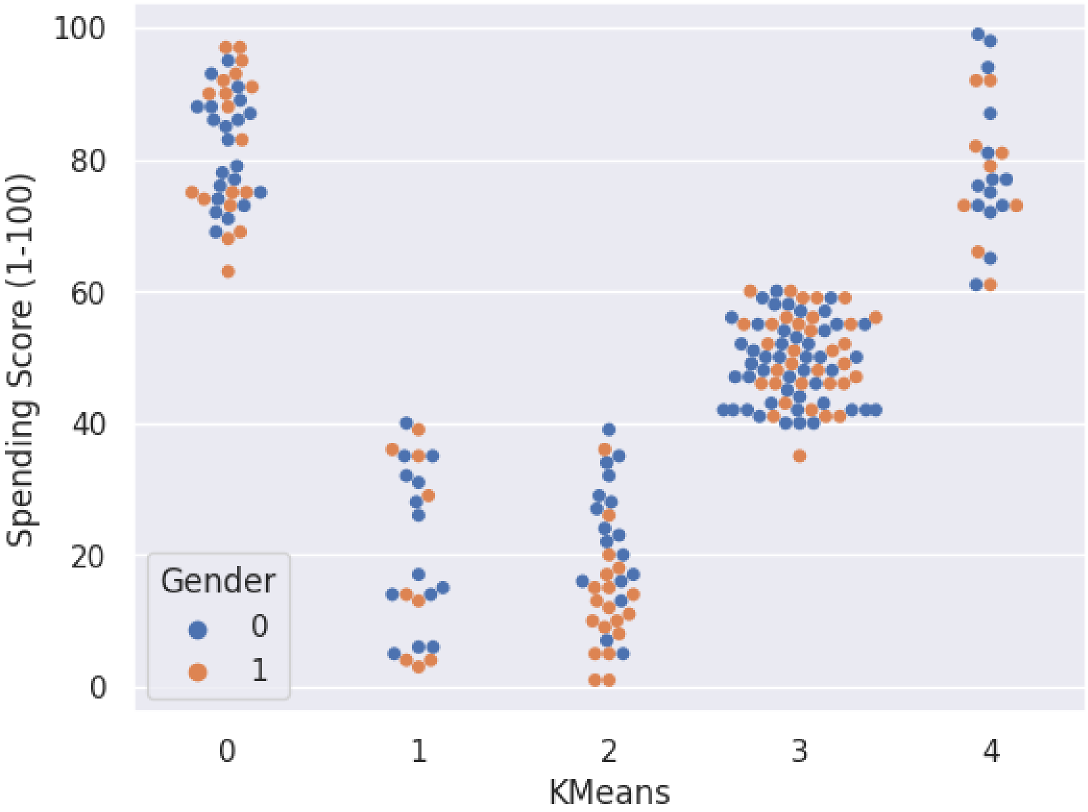
```
[38]: sns.jointplot(data=data,x="KMeans", y="Spending Score (1-100)",kind="kde")
```

```
[38]: <seaborn.axisgrid.JointGrid at 0x7d602fd1e3e0>
```



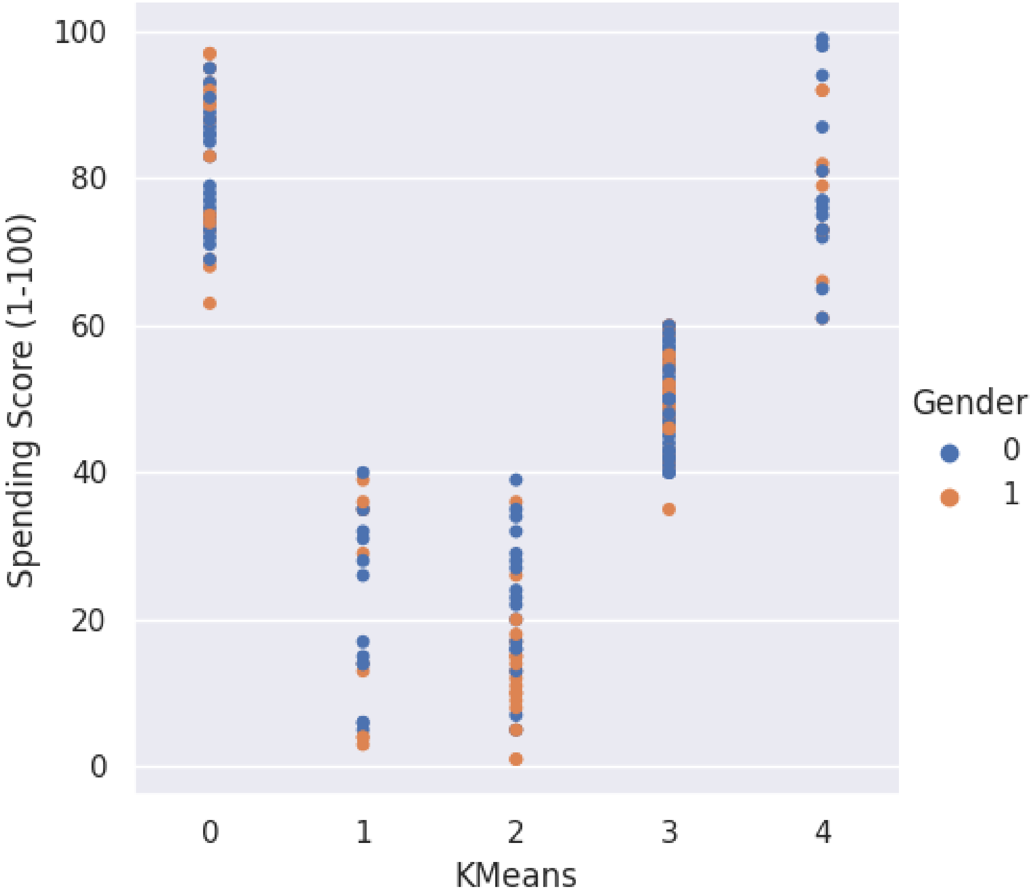
[40]:
```
sns.swarmplot(data=data, x="KMeans", y="Spending Score (1-100)",hue='Gender')
```

[40]:
```
<Axes: xlabel='KMeans', ylabel='Spending Score (1-100)'>
```



```
[41]: sns.catplot(data=data, x="KMeans", y="Spending Score (1-100)",␣
        ↪jitter=False,hue='Gender')
```

```
[41]: <seaborn.axisgrid.FacetGrid at 0x7d602e03f8e0>
```


100
80
60
40
20
0
Spending Score (1-100)
0
1
2
3
4
KMeans
Gender
0
1

```
[ ]:
```

```
[ ]:
```