group4-assign2-2

February 28, 2024

1 Introduction to Data Mining

1.1 Assignment2

1.2 Build and Implement K-means Clustering Algorithm

1.2.1 What Is Clustering?

Clustering is a set of techniques used to partition data into groups or clusters. Clusters are loosely defined as groups of data objects that are more similar to other objects in their cluster than they are to data objects in other clusters. In practice, clustering helps identify two qualities of data: - Meaningfulness: Meaningful clusters expand domain knowledge. For example, researchers applied clustering to gene expression experiments in the medical field. The clustering results identified groups of patients who respond differently to medical treatments. - Usefulness: Useful clusters, on the other hand, serve as an intermediate step in a data pipeline. For example, businesses use clustering for customer segmentation. The clustering results segment customers into groups with similar purchase histories, which businesses can use to create targeted advertising campaigns.

There are many other applications of clustering, such as document clustering and social network analysis. These applications are relevant in nearly every industry, making clustering a valuable skill for professionals working with data in any field.

[28]:
```
# Loading neccesary libraries:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier, export_graphviz, plot_tree
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans
from sklearn import preprocessing
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.metrics import pairwise
from sklearn.metrics import classification_report,confusion_matrix
```

```
from sklearn.metrics import accuracy_score # Performance measure - Accuracy
from sklearn.preprocessing import StandardScaler
import scipy.stats as stats
from scipy.stats import zscore
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
```

[29]:
```
import warnings
warnings.filterwarnings('ignore')
from pandas.plotting import parallel_coordinates
import matplotlib.pyplot as plt
plt.style.use('default')
```

[30]:
```
%matplotlib inline
# without this the plots would be opened  in a new window (not browser)
# with this instruction plots will be included in the notebook
```

[31]:
```
# Use %config InlineBackend.figure_format = 'retina'
# after %matplotlib inline to render higher resolution images
%config InlineBackend.figure_format = 'retina'
```

[32]:
```
# If you wish to use Google colab, the following code will allow you to mount␣
 ↪your Google Drive. Otherwise, comment on the following lines.
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call
drive.mount("/content/gdrive", force_remount=True).
```

[33]:
```
# To print multiple outputs
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
# Set it to None to display all columns in the dataframe
pd.set_option('display.max_columns', None)
```

2 Import dataset

[34]:
```
#Reading the data from google drive
data=pd.read_csv('/content/gdrive/MyDrive/Colab Notebooks/sessions.csv')
#Print that data imported successfully
print("Data imported successfully")
```

```
Data imported successfully
```

3 View top 10 rows of dataset

[35]:
```
#Write your code here
data.head(10)
```

[35]:

	Home	Products	Search	Prod_A	Prod_B	Prod_C	Cart	Purchase
0	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	1	0
2	1	0	0	0	0	0	0	0
3	1	1	1	1	0	0	1	1
4	1	0	1	1	1	0	1	1
5	1	1	1	0	1	0	0	0
6	1	0	1	0	0	0	1	1
7	1	0	1	0	1	0	0	0
8	1	1	1	0	1	0	1	0
9	1	0	1	1	1	1	1	1

4 Exploratory data analysis

5 View summary of Training dataset

[36]:
```
#Write your code here
data_train,data_test=train_test_split(data, test_size=0.25,random_state=7)
#print information
print("DataSet size {}".format(data.shape))
print("Train size {}".format(data_train.shape))
print("Test size {}".format(data_test.shape))
#summary of training data set
data_train.describe()
```

```
DataSet size (100, 8)
Train size (75, 8)
Test size (25, 8)
```

[36]:

	Home	Products	Search	Prod_A	Prod_B	Prod_C	\
count	75.000000	75.000000	75.000000	75.00000	75.00000	75.00000	
mean	0.586667	0.693333	0.426667	0.56000	0.56000	0.44000	
std	0.495748	0.464215	0.497924	0.49973	0.49973	0.49973	
min	0.000000	0.000000	0.000000	0.00000	0.00000	0.00000	
25%	0.000000	0.000000	0.000000	0.00000	0.00000	0.00000	
50%	1.000000	1.000000	0.000000	1.00000	1.00000	0.00000	
75%	1.000000	1.000000	1.000000	1.00000	1.00000	1.00000	
max	1.000000	1.000000	1.000000	1.00000	1.00000	1.00000	

	Cart	Purchase
count	75.000000	75.000000

```
mean    0.600000   0.400000
std     0.493197   0.493197
min     0.000000   0.000000
25%     0.000000   0.000000
50%     1.000000   0.000000
75%     1.000000   1.000000
max     1.000000   1.000000
```

Step8. Implement the K-means clustering algorithm to cluster these user sessions into segments. Try different clustering runs with various numbers of clusters (e.g., between 2 and 15), and select the result set(s) that seem to best answer as many of the following sub-questions as possible

- Elbow method is effective to find out optimal number of clusters for K-Means. This includes running K-means for range of values (Here range will be defined from 2 to 15)

[37]:
```
#Write your code here
#create dataframe
df=pd.DataFrame(data_train)
#different clustering runs with various numbers of clusters (e.g., between 2
 ↪and 15)
SSD=[] #within cluster sum of squares

for i in range(2,15):
  kmeans=KMeans(n_clusters=i,init='k-means++') #init=initializing centroid
  kmeans.fit(df)
  SSD.append(kmeans.inertia_)
plt.plot(range(2,15),SSD,'o-')
plt.xlabel("Number of clusters")
plt.ylabel("WCSS")
plt.grid(True)
plt.show()
```

[37]: KMeans(n_clusters=2)

[37]: KMeans(n_clusters=3)

[37]: KMeans(n_clusters=4)

[37]: KMeans(n_clusters=5)

[37]: KMeans(n_clusters=6)

[37]: KMeans(n_clusters=7)

[37]: KMeans()

[37]: KMeans(n_clusters=9)

[37]: KMeans(n_clusters=10)

[37]: KMeans(n_clusters=11)

[37]: KMeans(n_clusters=12)

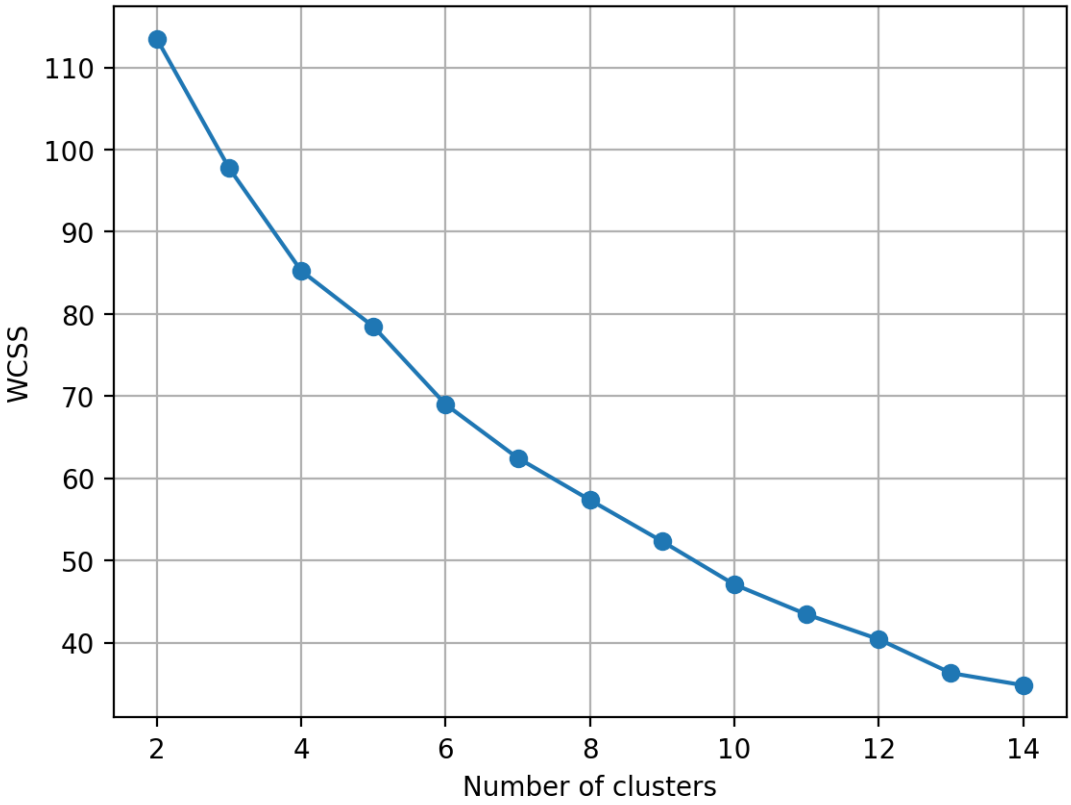
[37]: KMeans(n_clusters=13)

[37]: KMeans(n_clusters=14)

[37]: [<matplotlib.lines.Line2D at 0x7c58893df400>]

[37]: Text(0.5, 0, 'Number of clusters')

[37]: Text(0, 0.5, 'WCSS')



[38]:
```
# number of cluster =4 will be best to answer sub questions
```

Step9. If a new user is observed to access the following pages: Home => Search => Prod_B, according to your clusters, what other product should be recommended to this user? What if the new user has accessed the following sequence instead: Products => Prod_C? Explain your

answer based on your clustering results. ### >>>>>Answer and Show the result of this step here<<<<<

Now we know that optimal number for clusters is 4 we will use label_ which will assign data points to one of the clusters

[39]:
```
#For user sequence Home => Search => Prod_B,
df=pd.DataFrame(data_train)
#standardize columns
scaler=StandardScaler()
features_z=scaler.fit_transform(df)
#now we know clusters=4
kmeans = KMeans(n_clusters=4, random_state=42)
kmeans.fit(features_z)
```

[39]:
```
KMeans(n_clusters=4, random_state=42)
```

[40]:
```
df['Cluster']=kmeans.labels_
result=df[(df['Home']==1) & (df['Search']==1) & (df['Prod_B']==1)]
result
#if we want to visualize result
```

[40]:

	Home	Products	Search	Prod_A	Prod_B	Prod_C	Cart	Purchase	Cluster
33	1	1	1	0	1	1	0	0	1
10	1	0	1	1	1	1	1	0	1
28	1	1	1	1	1	1	0	0	1
35	1	1	1	1	1	1	0	0	1
9	1	0	1	1	1	1	1	1	3
4	1	0	1	1	1	0	1	1	3
7	1	0	1	0	1	0	0	0	1
8	1	1	1	0	1	0	1	0	1
14	1	0	1	1	1	1	0	0	1
25	1	1	1	0	1	1	0	0	1

According to the Clusters formed by the KNN Algorithm, any Customer who follows the order Home => Search => Prod_B: Fall into Cluster 2 (Majority) and in Cluster 4, so we can therefore recommend Prod_C and then followed by Prod_A.

[41]:
```
#Same for the user sequence Products => Prod_C
result=df[(df['Home']==0)&(df['Search']==0)&(df['Products']==1)&(df['Prod_C']==1)]
result.head(10)
```

[41]:

	Home	Products	Search	Prod_A	Prod_B	Prod_C	Cart	Purchase	Cluster
97	0	1	0	0	1	1	1	1	0
96	0	1	0	0	1	1	1	1	0
41	0	1	0	0	1	1	0	0	0
95	0	1	0	0	1	1	0	0	0
84	0	1	0	0	1	1	0	0	0

87	0	1	0	0	1	1	1	1	0
90	0	1	0	0	1	1	1	1	0
98	0	1	0	0	1	1	0	0	0
94	0	1	0	0	1	1	1	0	0
83	0	1	0	0	1	1	1	1	0

According to the Clusters formed by the KNN Algorithm, any Customer who follows the order Products => Prod_C: Fall into Cluster 1, so we can therefore recommend Prod_B.

Step10. Can clustering help us identify casual browsers (**"window shoppers"**), focused browsers (those who seem to know what products they are looking for), and searchers (those using the search function to find items they want)? If so, Do any of these groups show a higher or lower propensity to purchase? ### >>>>>Answer and Show the result of this step here<<<<<

[42]:
```
#Write your code here
#Reducing the number of cluster to 3, as we have only three customer segments
df1= df.copy()


features = df1[['Home', 'Products', 'Search']]
kmeans1 = KMeans(n_clusters=3, random_state=42)
df1['Cluster'] = kmeans1.fit_predict(features)
```

[43]:
```
home_distribution = df1.groupby('Cluster')['Home'].mean()
print(home_distribution)
```

```
Cluster
0    1.0
1    0.0
2    1.0
Name: Home, dtype: float64
```

[44]:
```
df1[df1['Cluster']==0].count()
```

[44]:
```
Home        23
Products    23
Search      23
Prod_A      23
Prod_B      23
Prod_C      23
Cart        23
Purchase    23
Cluster     23
dtype: int64
```

[45]:
```
Product_distribution = df1.groupby('Cluster')['Products'].mean()
print(Product_distribution)
```

```
Cluster
0    0.521739
1    0.741935
2    0.809524
Name: Products, dtype: float64
```

[46]:
```
df1[df1['Cluster']==2].count()
```

[46]:
```
Home        21
Products    21
Search      21
Prod_A      21
Prod_B      21
Prod_C      21
Cart        21
Purchase    21
Cluster     21
dtype: int64
```

[47]:
```
df1[df1['Cluster']==1].count()
```

[47]:
```
Home        31
Products    31
Search      31
Prod_A      31
Prod_B      31
Prod_C      31
Cart        31
Purchase    31
Cluster     31
dtype: int64
```

[48]:
```
average_purchase_by_cluster = df1.groupby('Cluster')['Purchase'].mean()

print(average_purchase_by_cluster)
```

```
Cluster
0    0.391304
1    0.419355
2    0.380952
Name: Purchase, dtype: float64
```

From The above analysis we can conclude that , And the number of focused

- The number of window shoppers is 30.66% of total Shoppers
- The number of Focused Shoppers is 28% of total shoppers
- And the Rest is Searchers

But as the cluster overlaps and is not perfectly separated, thus a window shopper is highly likely to be a searcher as the shopper may navigate from home to search, Rest Aside the Clustering also tells that **Focused Shoppers** are the ones who purchased a product.

Step11. Do any clusters/segments show particular interest in one or more products? If so, can we identify any unique characteristics of their navigational behaviour or their purchase propensity? ### >>>>>Answer and Show the result of this step here<<<<<

[49]:
```
#Write your code here
#For Checking interset
product_interest_by_cluster = df1.groupby('Cluster')[['Prod_A', 'Prod_B',␣
 ↪'Prod_C']].mean()
print(product_interest_by_cluster)

print("-------------------------------------------------------")

#To Examine the navigational behavior
navigational_behavior_by_cluster = df.groupby('Cluster')[['Home', 'Products',␣
 ↪'Search', 'Cart']].mean()
print(navigational_behavior_by_cluster)
```

```
           Prod_A    Prod_B    Prod_C
Cluster
0        0.521739  0.434783  0.521739
1        0.354839  0.806452  0.483871
2        0.904762  0.333333  0.285714
-------------------------------------------------------
              Home  Products    Search      Cart
Cluster
0        0.000000  1.000000  0.350000  0.500000
1        1.000000  0.500000  0.928571  0.214286
2        0.722222  0.722222  0.111111  0.500000
3        0.739130  0.521739  0.434783  1.000000
```

Step12. Suppose we know that, during data collection, independent banner ads were placed on some popular sites pointing to products A and B. Can we identify segments corresponding to visitors that respond to the ads? (note that such users are likely to enter the site by going directly to product pages rather than navigating from the Home page). If so, can we determine if either of these promotional campaigns are having any success? ### >>>>>Answer and Show the result of this step here<<<<<

Note: For this part, you should submit your clustering result summary (including the cluster centroids), the final data set, which shows the final assignment of these sessions to clusters, and your answers to the above questions, along with your justification based on the clustering results. Also, you should show the cluster visualization to identify interesting distributions of various page visits among and within clusters.

[50]:
```
import pandas as pd
import seaborn as sns
```

```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt


features = df.drop('Cluster', axis=1)

pca = PCA(n_components=2, random_state=42)
pca_result = pca.fit_transform(features)

pca_df = pd.DataFrame(pca_result, columns=['Principal Component 1', 'Principal␣
 ↪Component 2'])
pca_df['Cluster'] = df['Cluster']

plt.figure(figsize=(10, 6))
sns.scatterplot(x='Principal Component 1', y='Principal Component 2',␣
 ↪hue='Cluster', data=pca_df, palette='viridis', s=100, alpha=0.7)

plt.title('Scatter Plot of Customer Behavior with Clusters (PCA)')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')

plt.show()
```

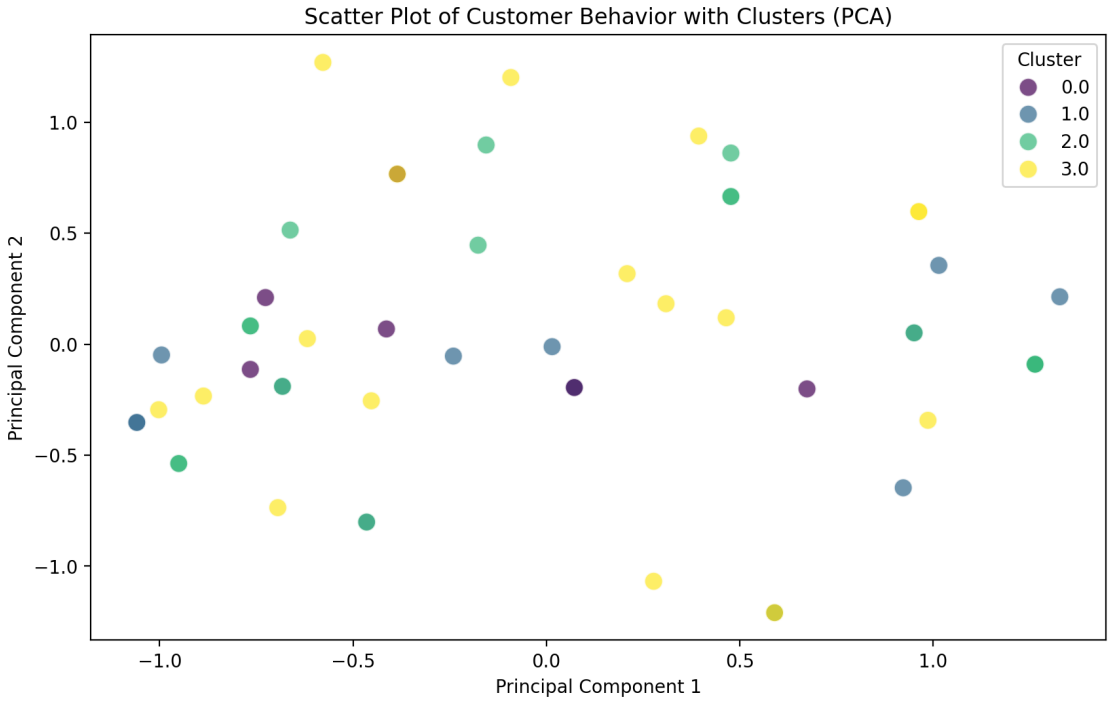
[50]: <Figure size 1000x600 with 0 Axes>

[50]: <Axes: xlabel='Principal Component 1', ylabel='Principal Component 2'>

[50]: Text(0.5, 1.0, 'Scatter Plot of Customer Behavior with Clusters (PCA)')

[50]: Text(0.5, 0, 'Principal Component 1')

[50]: Text(0, 0.5, 'Principal Component 2')



##References:

Farid Mammadaliyev. (2020, August 21). Determining optimal K in K-Means Clustering using Elbow method [Video]. YouTube. https://www.youtube.com/watch?v=0h4yhFtoXOU

Andy McDonald. (2021, November 17). K-Means Clustering Algorithm with Python Tutorial [Video]. YouTube. https://www.youtube.com/watch?v=iNlZ3IU5Ffw

Saji, B. (2023, September 20). Elbow method for finding the optimal number of clusters in K-Means. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clustering-algorithm-in-machine-learning/