Digital Learning Technology Final Project

611221201

Dec 2023

1 Introduction

For the final project,we will use the csv file to implement clustering by KMeans algorithm,eventually we will use plt library to display the clustering result .

2 Code process

2.1 Import library

- **Pandas**: Data cleaning and analysis.
- **Numpy**: Handle large multidimensional arrays and matrices.
- **Matplotlib**: Creating static, animated, and interactive visualizations.
- **Seaborn**: Data visualization library based on matplotlib.
- **Sklearn**: Machine learning library.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
```

Figure 1: Import library

2.2 Read csv file

In order to read csv file,we use pandas library to read csv file and store it in a variable called data.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
```

```
data = pd.read_csv('Mall_Customers.csv')
data
```

	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 × 5 columns

Figure 2: Read csv file

2.3 Show csv information

Showing what columns, size and data type are there

```
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
```

Figure 3: csv information

2.4 Show csv basic situation

View the basic situation of continuous value data in DataFrame

```
data.describe()
```

	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

Figure 4: csv basic situation

2.5 Encode the Gender

Encode the gender column in the data frame,Converting text data into numbers can make the data more standardized and easier for algorithm processing

```
la = LabelEncoder()
data.Gender = la.fit_transform(data.Gender)
data
```

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

200 rows × 5 columns

Figure 5: Encode the Gender

2.6 Perform K-means clustering

The KMeans class from the scikit-learn library is used to perform K-means clustering, and the dataset data is being fitted to this model.

```
model = KMeans(n_clusters = 5)
model.fit(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(
KMeans(n_clusters=5)
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.
```

Figure 6: K-means clustering

2.7 Nearest cluster center distances

Refers to the inertia attribute of the K-means clustering model. In K-means clustering, inertia is used to measure the sum of the distances of all the data points to their nearest cluster center. Specifically, it calculates the sum of

the squared distances between each data point and the center of the cluster it has been assigned to

```
model.inertia_
```

```
157505.72072477575
```

Figure 7: nearest cluster center

2.8 K-means Clustering Experiment

1. **Initialization of Two Empty Lists** *clustersn* and *j*:
   These lists are used to store the varying numbers of clusters and their corresponding inertia values.

2. **Looping Through K-means Clustering**:

   - The process involves a `for` loop, iterating the number of clusters from 1 to 11.
   - In each iteration:
     (a) A `KMeans` instance is created with `n_clusters` $= i$, where i denotes the number of clusters.
     (b) The model is fitted to the dataset `data` using `model.fit(data)`, effectively performing clustering with the given number of clusters.
     (c) The current cluster number i is appended to the list *clustersn*.
     (d) The inertia value of the current model `model.inertia_` is appended to the list *j*.

3. **Creating and Displaying a DataFrame**:

   - A new DataFrame is created using Pandas, comprising two columns: `clusters` (the number of clusters) and `j` (the corresponding inertia values).

- This DataFrame aids in visualizing the impact of different cluster numbers on the clustering effect (measured by inertia values).

4. **Determining the Optimal Number of Clusters**:

   - By observing the inertia values for different cluster numbers, one can determine the optimal number of clusters.
   - Generally, as the number of clusters increases, the inertia value decreases, but too many clusters might lead to overfitting.
   - Hence, selecting a cluster number where the decrease in inertia begins to slow down is often a good choice; this point is known as the "elbow" point.
   - This method is very practical in choosing the optimal number of clusters for K-means clustering.

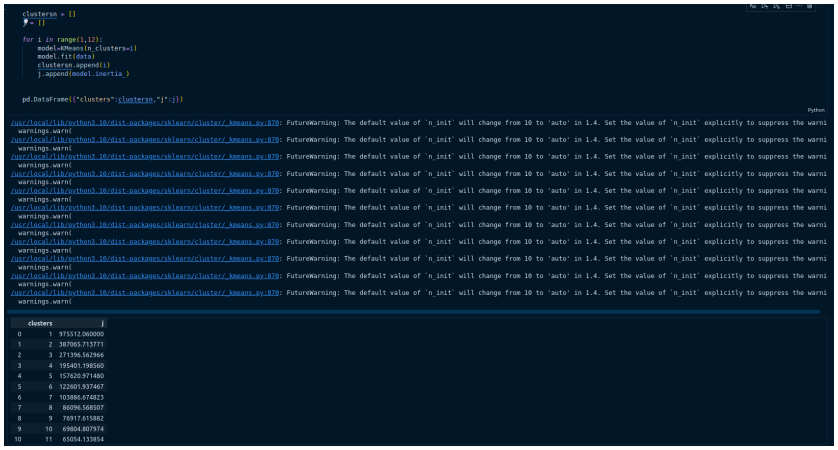

Figure 8: K-means Clustering Experiment

2.9 cluster a dataset

1. **Creating a KMeans Instance**: The line `model = KMeans(n_clusters = 4)` creates an instance of the KMeans class with `n_clusters = 4`, meaning that the algorithm will divide the data into 4 clusters (or categories).

2. **Fitting the Model**: By executing `model.fit(data)`, the model is fitted to the dataset `data`. During this step, the algorithm performs the following actions:

   - Chooses 4 initial cluster centers.

- Assigns each data point to the closest cluster center.
- Updates the cluster centers to be the average location of the points assigned to them.
- Repeats the above steps until the cluster centers no longer change, or other stopping criteria are met.

3. **Prediction**: The line `pre = model.predict(data)` uses the fitted model to predict the cluster assignments for the same dataset `data`, determining which cluster each data point belongs to. The `predict` method returns an array where each element represents the index of the cluster to which the corresponding data point has been assigned.

4. **Result**: `pre` contains the cluster assignment for each data point. If you print `pre`, it will display a one-dimensional array, with each value indicating the number of the cluster to which the data point has been assigned, ranging from 0 to 3.

```
model = KMeans(n_clusters = 4)
model.fit(data)
pre = model.predict(data)
pre
```

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

```
array([3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 3, 0, 0, 3, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 2, 1, 2, 1, 2,
       1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2,
       1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2,
       1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2,
       1, 2], dtype=int32)
```

Figure 9: Cluster a dataset

2.10 Add prediction result to data frame

The original dataset data is extended to include not just the original features but also a column representing the clustering results for each data point

```
data['k_mean'] = pre
data
```

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

200 rows × 6 columns

Figure 10: Predict result

2.11 Split to four subset

```
df1 = data[data['k_mean'] == 0]
df2 = data[data['k_mean'] == 1]
df3 = data[data['k_mean'] == 2]
df4 = data[data['k_mean'] == 3]
```

```
df1
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)	k_mean
57	58	1	69	44	46	0
59	60	1	53	46	46	0
60	61	1	70	46	56	0
62	63	0	67	47	52	0
63	64	0	54	47	59	0
...	...	...	...	...	...	...
120	121	1	27	67	56	0
121	122	0	38	67	40	0
122	123	0	40	69	58	0
124	125	0	23	70	29	0
126	127	1	43	71	35	0

66 rows × 6 columns

Figure 11: Split to four subset

3 Visualization of clustering results

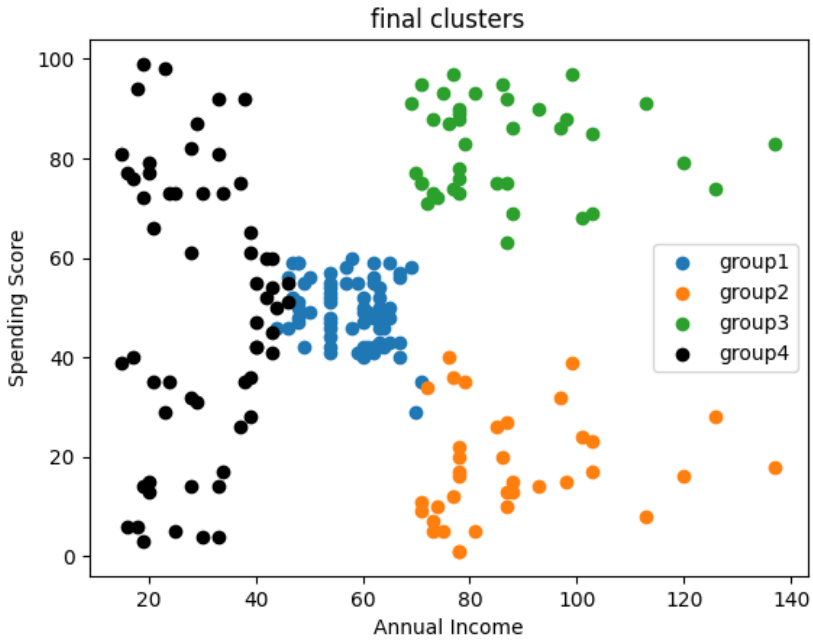

Figure 12: Clustering results

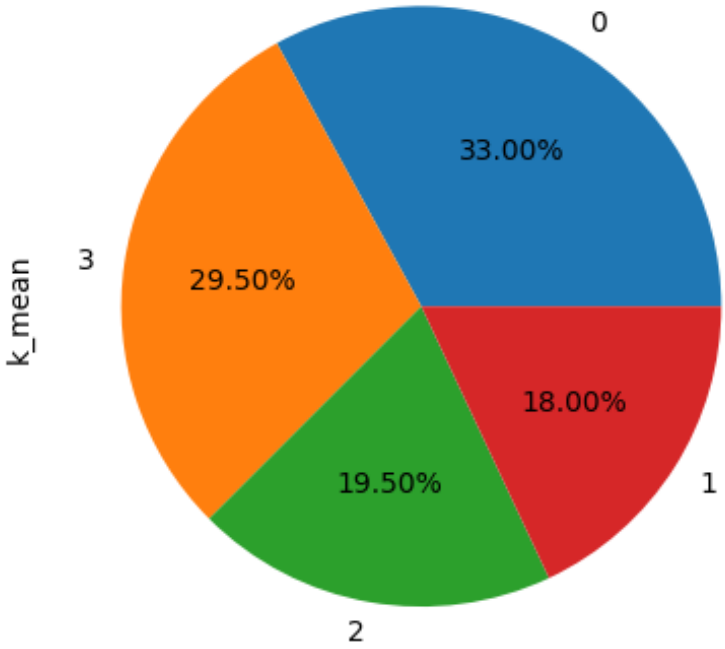

Figure 13: Distribution of different clusters

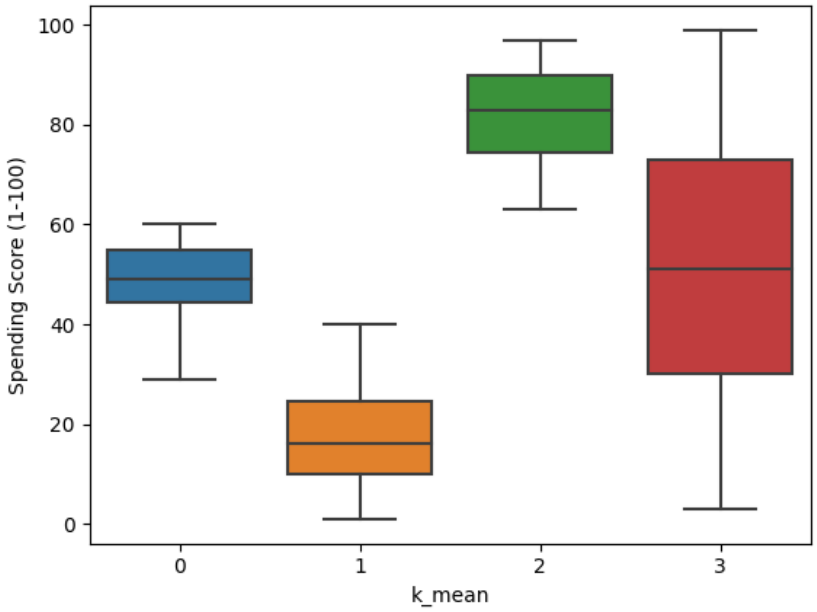

Figure 14: Consumption score distribution of different clusters based on K-means clustering results

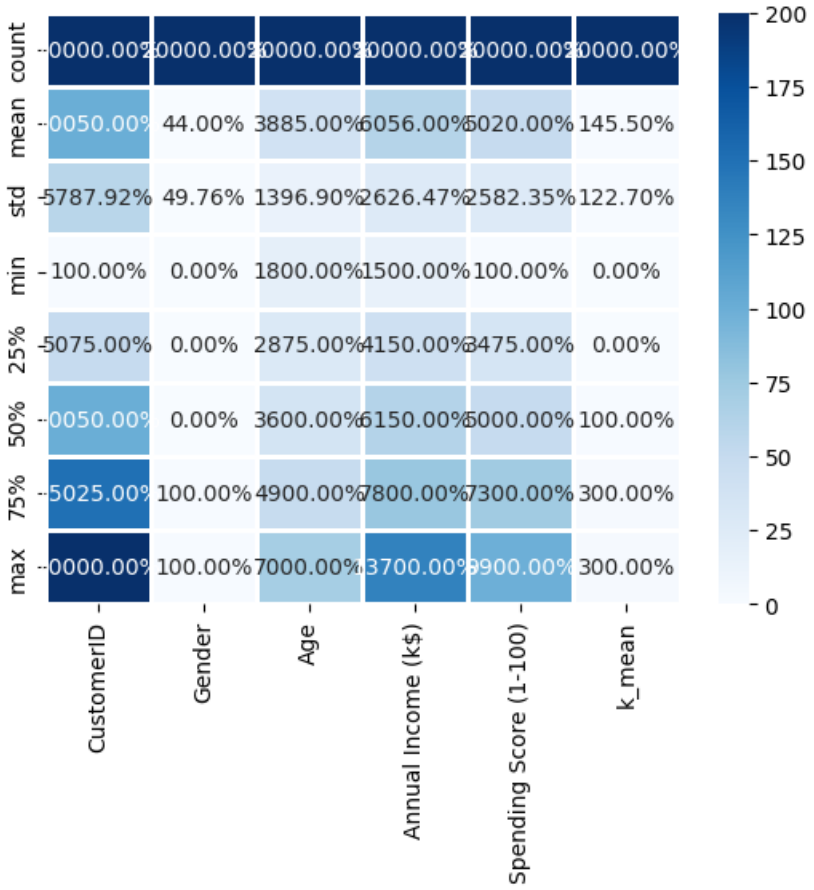

Figure 15: Distribution and relative size across different features

Figure 16: Relationship between annual income and consumption score