

ML Project Report on

MALL CUSTOMER SEGMENTATION

INDEX

1. Problem Statement
2. Abstract
3. Introduction
4. Data Set Information
5. Code and Output
6. Conclusion and Future Scope
7. References

PROBLEM STATEMENT

To develop a machine learning model that can identify and group customers based on their shopping patterns and behavior. The model should use customer data such as age, gender, annual income, and spending score to segment customers into different groups. This segmentation will enable mall owners to target their marketing efforts more effectively, improve customer experience, and optimize their business operations.

ABSTRACT

Mall customer segmentation refers to the process of grouping customers based on their shopping patterns and behavior in a mall. This is achieved by analyzing customer data, including demographic information such as age, gender, and income, and behavioral data such as purchase history and frequency.

The purpose of mall customer segmentation is to enable mall owners to better understand their customers and provide targeted marketing efforts that cater to their needs and preferences. By segmenting customers, mall owners can identify trends, patterns, and preferences that can be used to create more personalized shopping experiences and improve customer satisfaction. Additionally, mall owners can optimize their business operations by targeting specific customer segments with promotions, discounts, and other incentives.

Furthermore, mall customer segmentation can also help in identifying high-value customers who contribute significantly to the mall's revenue, allowing mall owners to focus on retaining and nurturing these customers. It can also assist in identifying customer groups that require more attention and support, such as first-time visitors or customers who have experienced issues during their shopping experience.

Overall, mall customer segmentation is an essential tool in the retail industry that allows mall owners to better understand their customers, optimize their operations, and provide personalized experiences that cater to the needs and preferences of their customers.

INTRODUCTION

Implemented a Mall Customer Segmentation based Machine Learning model which focuses on different clustering methods.

The clustering methods implemented are:

- 1) K means clustering
- 2) Mean-Shift Clustering
- 3) Hierarchical Clustering

Various types of graphs are plotted for comparing the results, namely:

- 1) Normal Scatter Plot
- 2) Plot with Clusters (K-means)
- 3) Distplot
- 4) Pie Chart
- 5) Bar Graph
- 6) Pair plot
- 7) Heatmap
- 8) Scatter Matrix
- 9) Elbow Method Plot


Based on the results obtained through these plots, we have compared the two clustering methods to find out which clustering method is more effective.

DATA SET INFORMATION:

The dataset initially consisted of 300 rows and was taken from kaggle, which was later updated to 500 rows manually.

Link for the dataset: [Mall_Customers.csv](#)

Dataset snippet

<div>  Mall_Customers ☆ 🔗 ☁ </div> <div> File Edit View Insert Format Data Tools Extensions Help </div>						
<div> ↶ ↷ 🖨 🔍 100% \$ % .0 .00 123 Defaul... 10 </div>						
A1	fx CustomerID					
	A	B	C	D	E	
1	CustomerID	Gender	Age	Annual Income (Spending Score (1	
2	1	Male	19	15	39	
3	2	Male	21	15	81	
4	3	Female	20	16	6	
5	4	Female	23	16	77	
6	5	Female	31	17	40	
7	6	Female	22	17	76	
8	7	Female	35	18	6	
9	8	Female	23	18	94	
10	9	Male	64	19	3	
11	10	Female	30	19	72	
12	11	Male	67	19	14	
13	12	Female	35	19	99	
14	13	Female	58	20	15	
15	14	Female	24	20	77	
16	15	Male	37	20	13	
17	16	Male	22	20	79	
18	17	Female	35	21	35	
19	18	Male	20	21	66	
20	19	Male	52	23	29	
21	20	Female	35	23	98	
22	21	Male	35	24	35	

More than 500 data are included.

CODE AND OUTPUT:

```
!pip install bokeh #data visualization library for creating interactive
plots, dashboards and applications.
!pip install scipy #used for scientific computing
```

```
!pip install pyngrok # allows you to expose a local web server to the
internet using ngrok, a secure tunneling service.
```

```
# IMPORTING THE LIBRARIES
# numpy and pandas for data manipulation
# matplotlib for data visualization
# sklearn.cluster KMeans for clustering analysis
```

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
```

```
# READING THE DATASET
```

```
import io
from google.colab import files
uploaded = files.upload()
df = pd.read_csv(io.BytesIO(uploaded["Mall_Customers.csv"]))
print("\nGIVEN DATASET : ")
print("\n-----\n"
)
print(df)
print("\n-----")
print()
```

```
GIVEN DATASET :
```

```
-----
```

	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
..
495	496	Male	64	10	64
496	497	Female	36	50	51

497	498	Male	61	16	95
498	499	Female	58	15	86
499	500	Male	25	12	84

[500 rows x 5 columns]

```
-----
# Displays first 10 rows of the dataframe
df.head(10)
```

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
5	6	Female	22	17 76
6	7	Female	35	18 6
7	8	Female	23	18 94
8	9	Male	64	19 3
9	10	Female	30	19 72

```
# Displays last a number of rows of the dataframe
a=int(input("Enter the number of rows: "))
df.tail(a)
```

Enter the number of rows: 5

CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)
495	496	Male	64	10 64
496	497	Female	36	50 51

497	498	Male	61	16	95
498	499	Female	58	15	86
499	500	Male	25	12	84

```
# The method returns a summary of statistics for each column in the
DataFrame
df.describe()
CustomerIDAgeAnnual Income (k$)Spending Score
(1-100)count500.000000500.000000500.000000500.000000mean250.49800039.412000
48.1980049.414000std144.47850413.62459225.3802924.865462min1.00000018.0000
003.000001.00000025%125.75000029.75000029.75000029.00000050%250.50000036.00
000043.00000048.00000075%375.25000049.00000063.00000068.250000max500.0000009
5.000000143.00000099.000000
# Prints information about the DataFrame
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 5 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   CustomerID                            500 non-null   int64
1   Gender                                500 non-null   object
2   Age                                    500 non-null   int64
3   Annual Income (k$)                    500 non-null   int64
4   Spending Score (1-100)                 500 non-null   int64
dtypes: int64(4), object(1)
memory usage: 19.7+ KB
# Checks whether any value in the DataFrame is null (NaN or None) and
returns a Boolean Series indicating which columns have missing values.
df.isnull().any()
CustomerID                False
Gender                    False
Age                       False
Annual Income (k$)        False
Spending Score (1-100)    False
dtype: bool
# Counts distinct gender
df['Gender'].value_counts()
Female      260
```

```

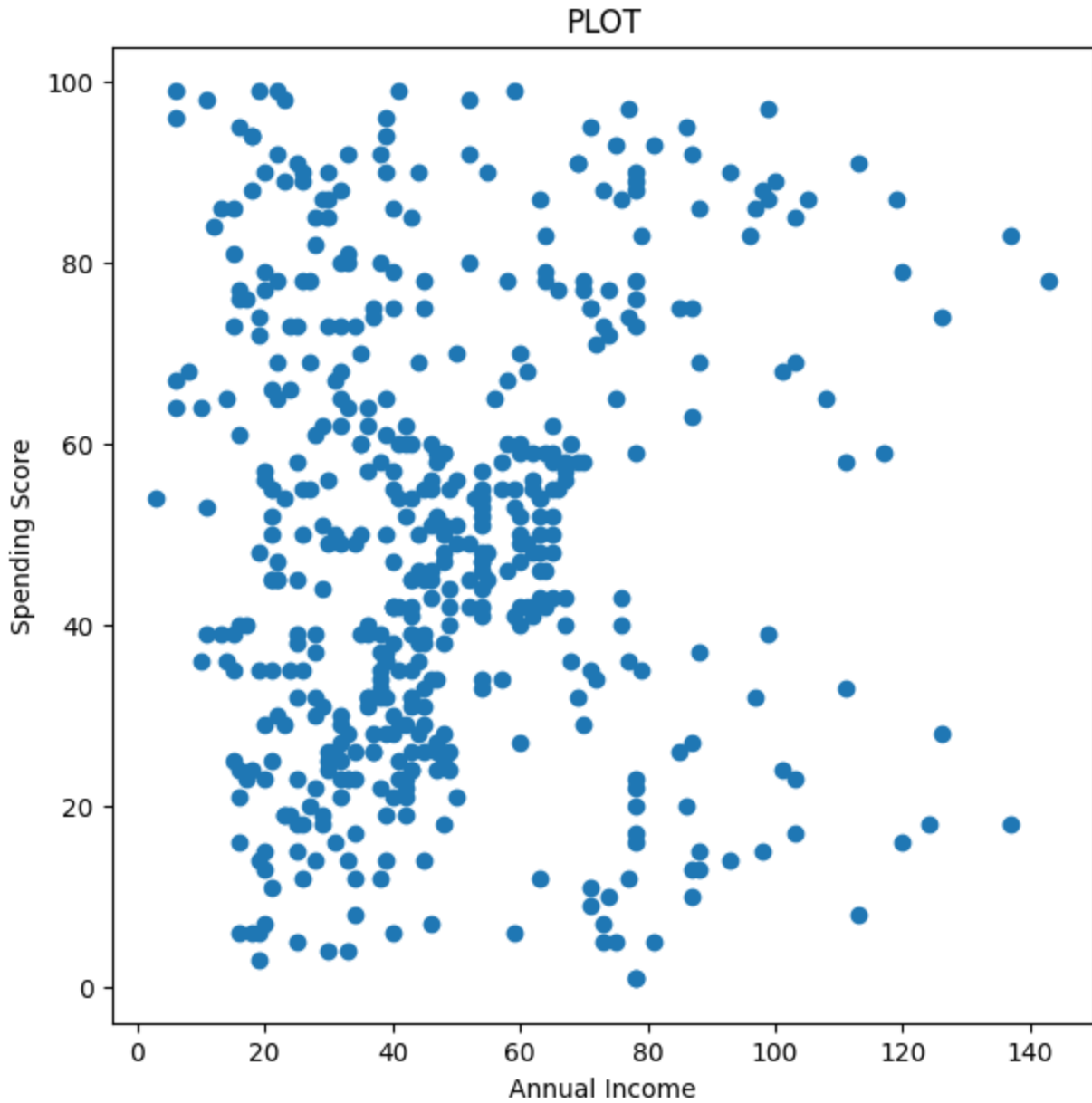
Male          240
Name: Gender, dtype: int64
# %time used to measure the execution time of reading a CSV file
Mall_Customers.csv into a Pandas DataFrame using pd.read_csv() function
print("\nEXECUTION TIME: ")
%time df = pd.read_csv('Mall_Customers.csv')

# Displays the dimensions of the DataFrame
print("\nThe shape of dataframe: ")
print(df.shape)
EXECUTION TIME:
CPU times: user 2.77 ms, sys: 0 ns, total: 2.77 ms
Wall time: 3.1 ms

The shape of dataframe:
(500, 5)
# Creates Numpy array x1 and stores data with all rows and 4th,5th columns
of the dataframe.
x1 = df.iloc[:, [3, 4]].values
# Normal Scatter plot
#scatter plots provide an effective way to visually represent and analyze
the relationship between two variables in a dataset.
print("\nSCATTER PLOT")
x = df["Annual Income (k$)"] # "Annual Income (k$)" value of 50, it means
that their annual income is $50,000.
y = df["Spending Score (1-100)"]

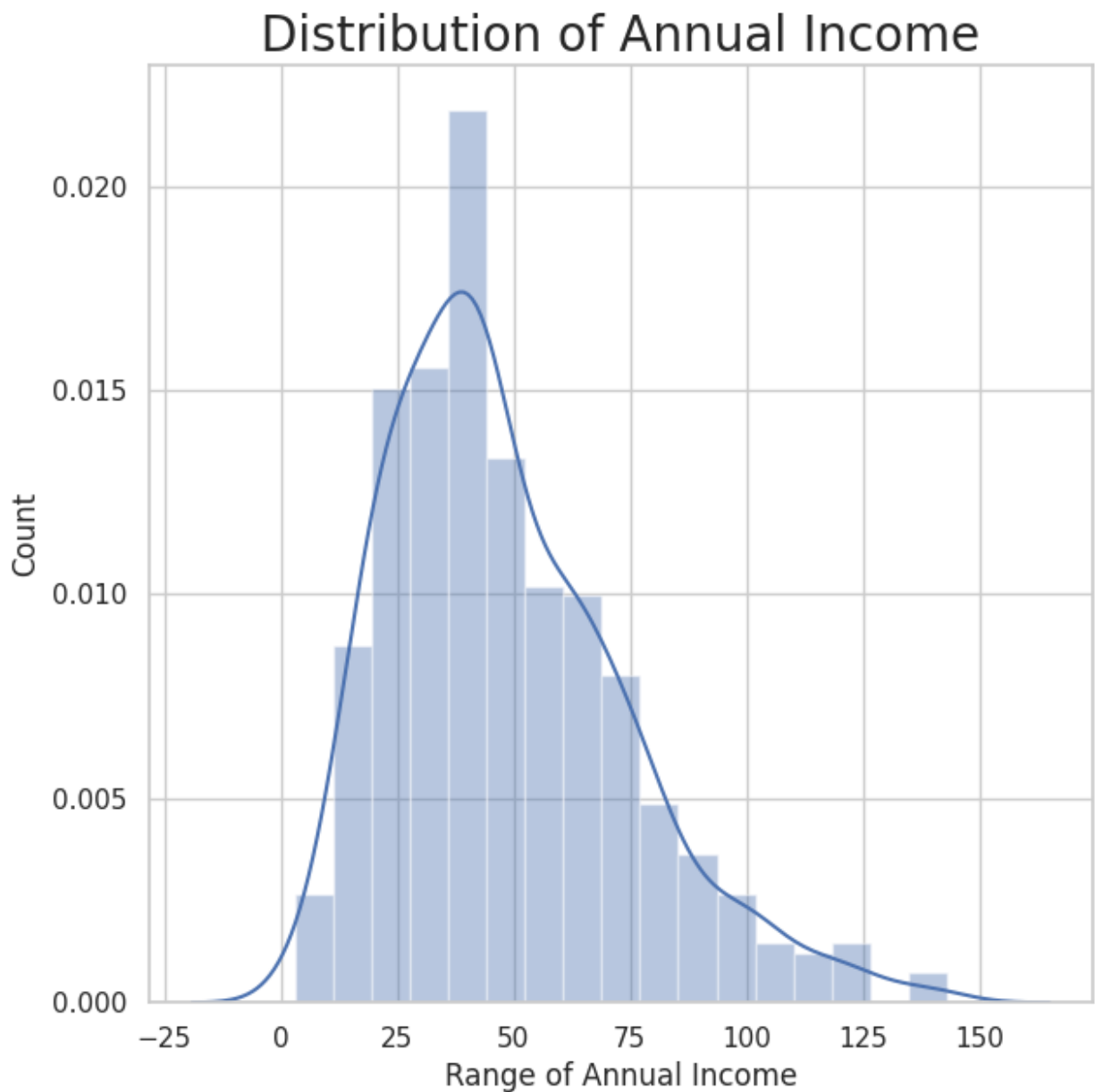
plt.scatter(x , y )
plt.xlabel("Annual Income")
plt.ylabel("Spending Score")
plt.title("PLOT")
plt.show()
SCATTER PLOT

```

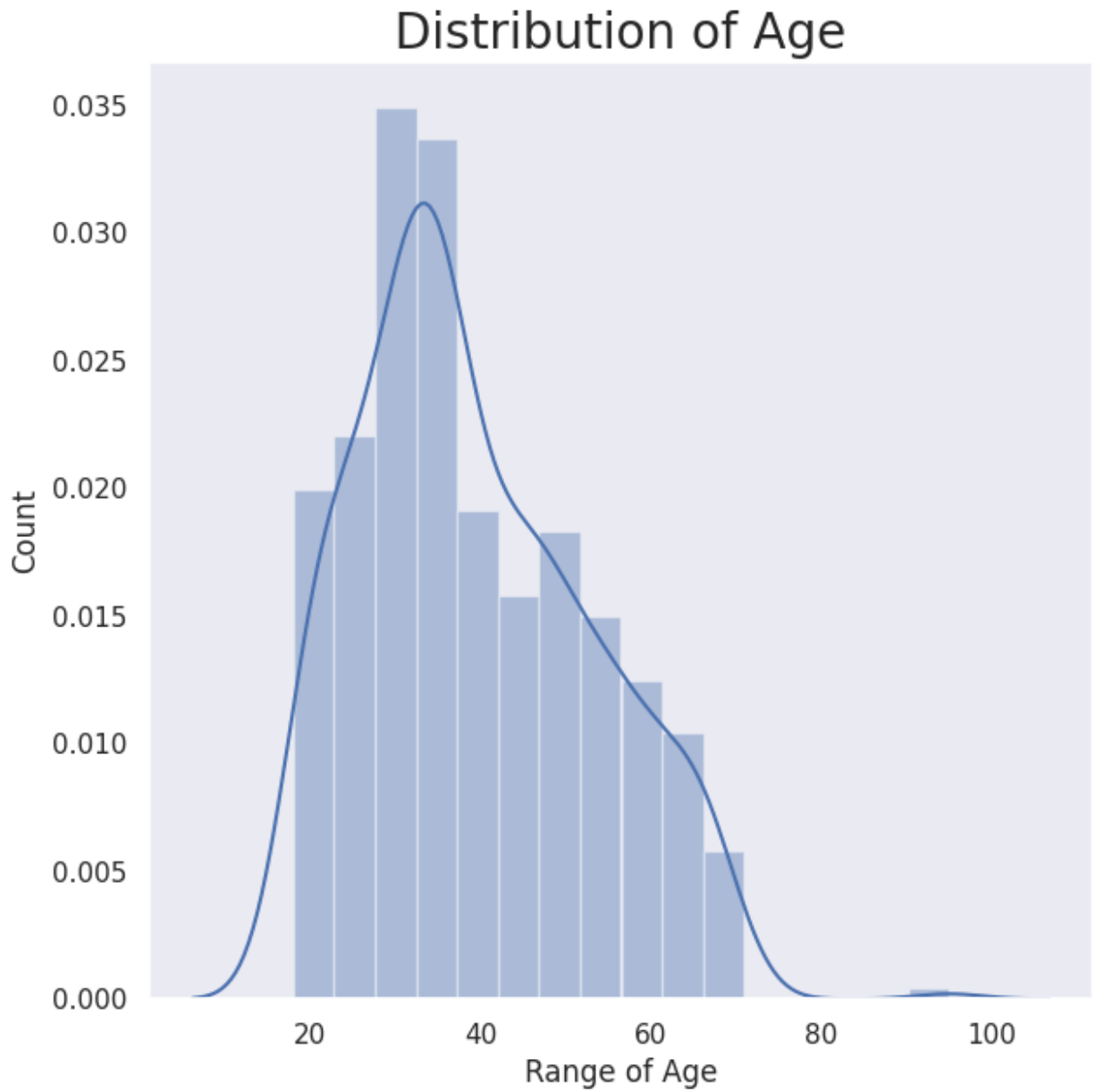


```
import seaborn as sns
#Seaborn is a Python data visualization library based on matplotlib that
provides a high-level interface for creating informative and attractive
statistical graphics.
sns.set(style = 'whitegrid') # sets the style of the plot to have a white
grid background.
sns.distplot(df['Annual Income (k$)']) # distplot() function plots a
univariate distribution of observations.
plt.title('Distribution of Annual Income', fontsize = 20)
plt.xlabel('Range of Annual Income')
```

```
plt.ylabel('Count') #the frequency or count of observations in each bin of
the histogram.
plt.show()
```



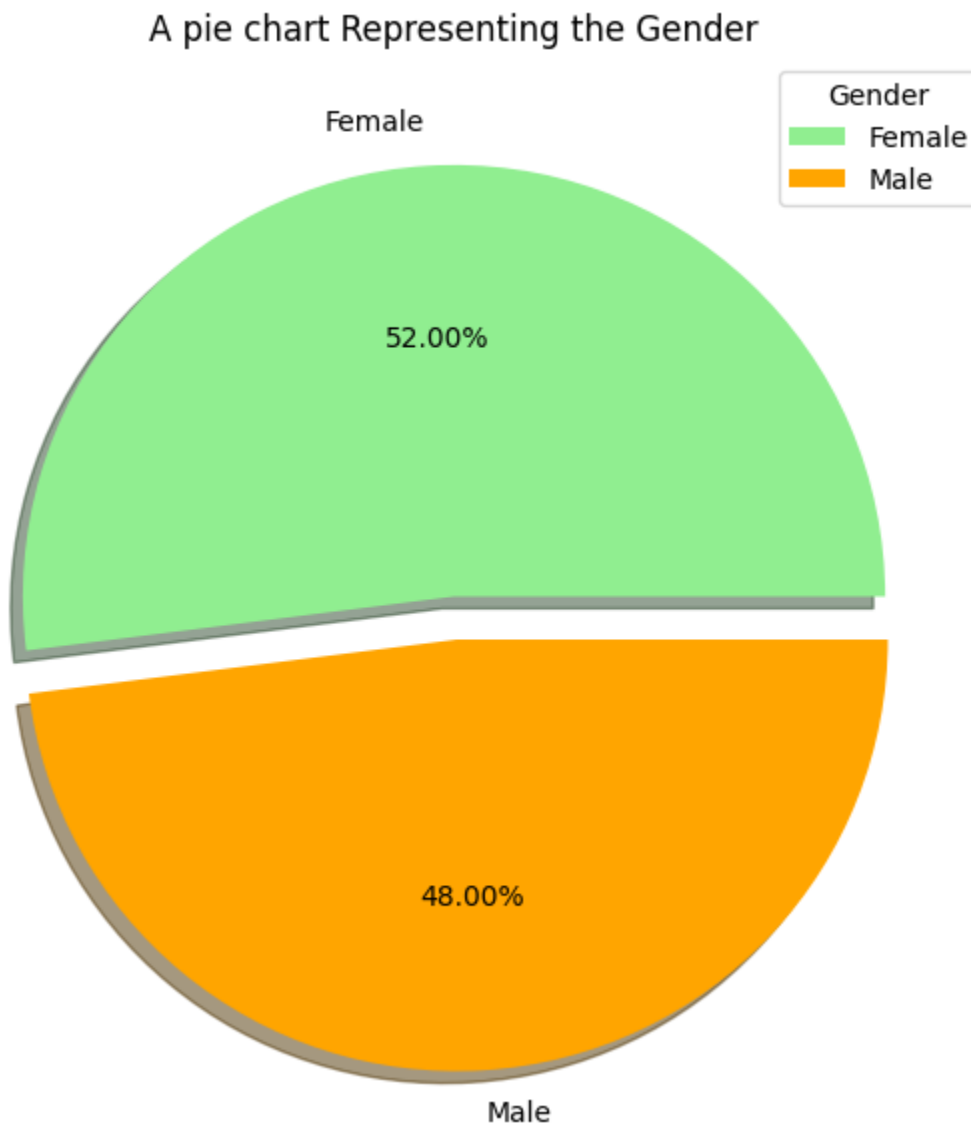
```
sns.set(style = 'dark') # sets style of plot to dark
sns.distplot(df['Age']) #creates histogram of the age variable using the
displot()
plt.title('Distribution of Age', fontsize = 20)
plt.xlabel('Range of Age')
plt.ylabel('Count')
plt.show()
```



```
labels = ['Female', 'Male'] #creates alist of 2 labels
size = [260, 240] # creates list of 2 values
colors = ['lightgreen', 'orange'] # sets colours for 2 sections of
pie-chart
explode = [0, 0.1] # sets the amount of expolde(distance from center) for
the second portion of pie chart

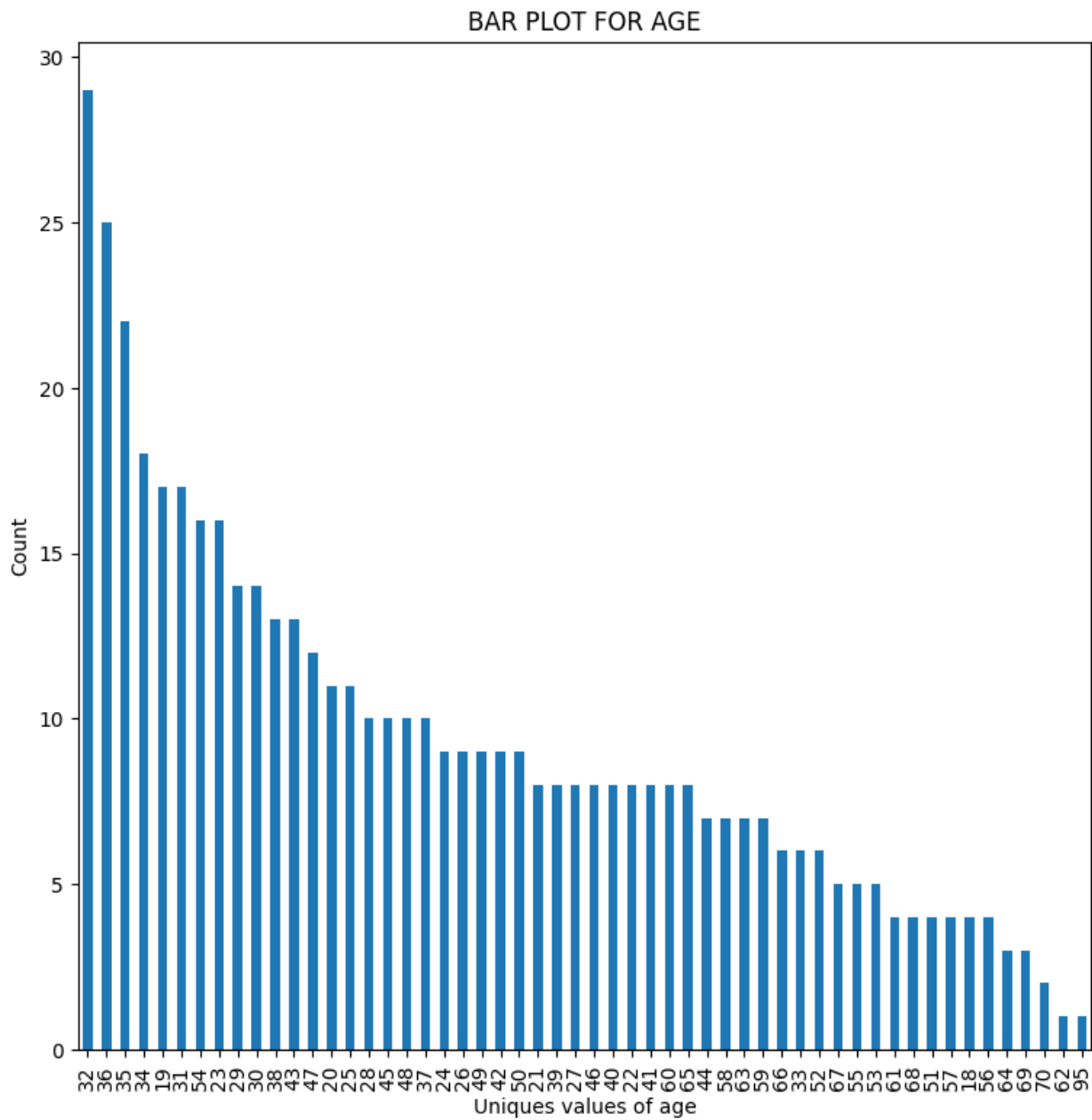
plt.rcParams['figure.figsize'] = (7, 7) #sets size of figure to 7 inches
by 7 inches
```

```
plt.pie(size, colors = colors, explode = explode, labels = labels, shadow
= True, autopct = '%.2f%%')
plt.title('A pie chart Representing the Gender')
plt.axis('off') # turns off axis of chart
plt.legend(title='Gender', loc='upper right')
plt.show()
```

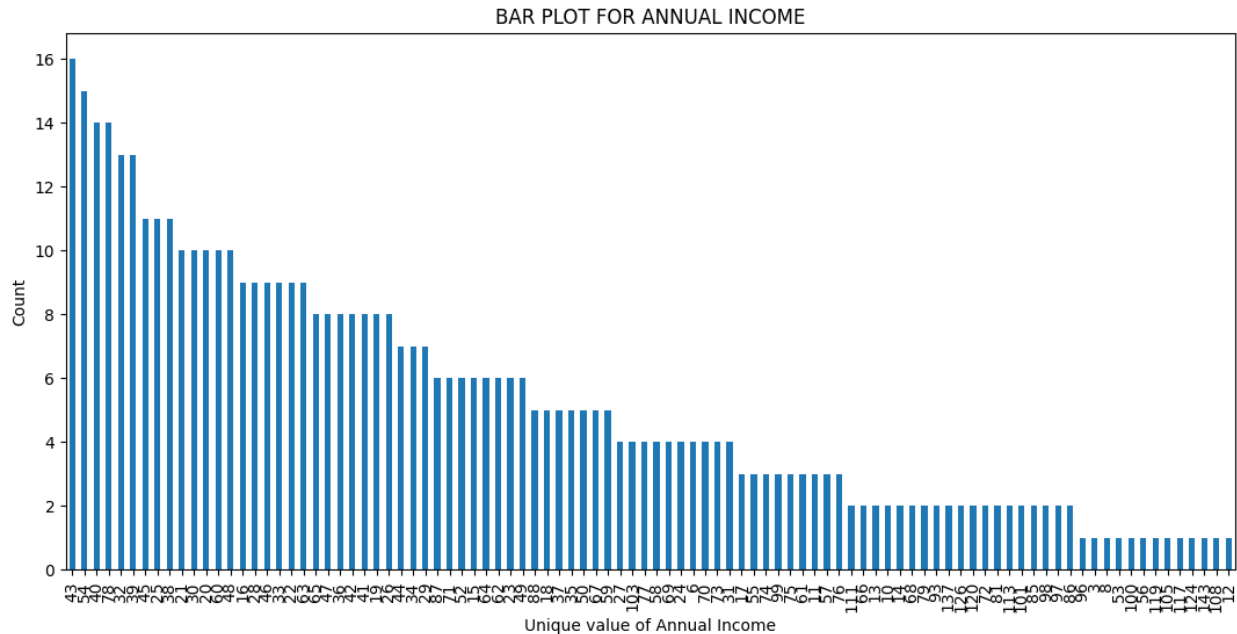


```
# Used to plot- bar plot which counts the occurrences of each unique value
in 'Age' column
df['Age'].value_counts().plot.bar(figsize = (9, 9))
plt.title("BAR PLOT FOR AGE")
plt.xlabel("Uniques values of age")
plt.ylabel("Count")
```

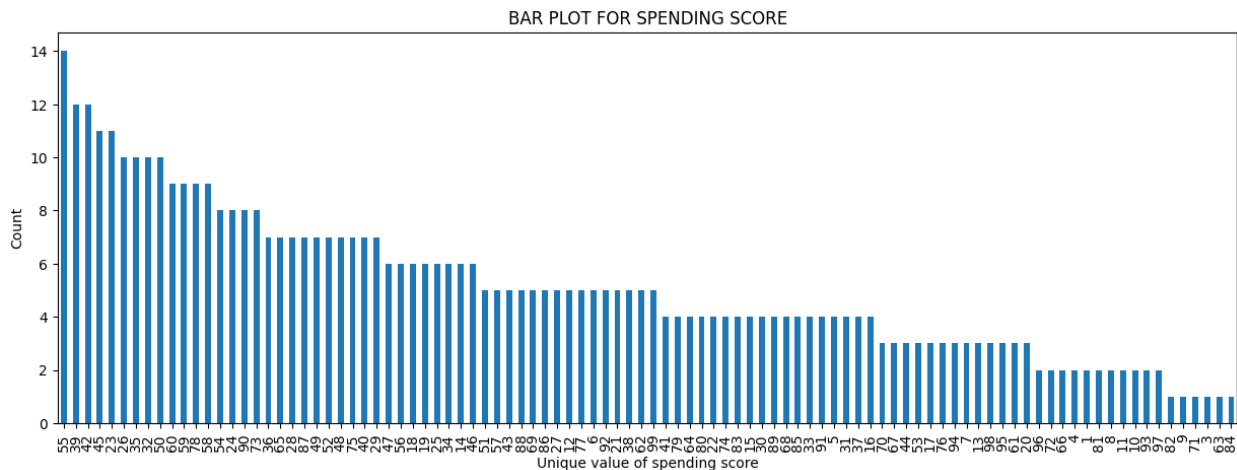
```
plt.show()
```



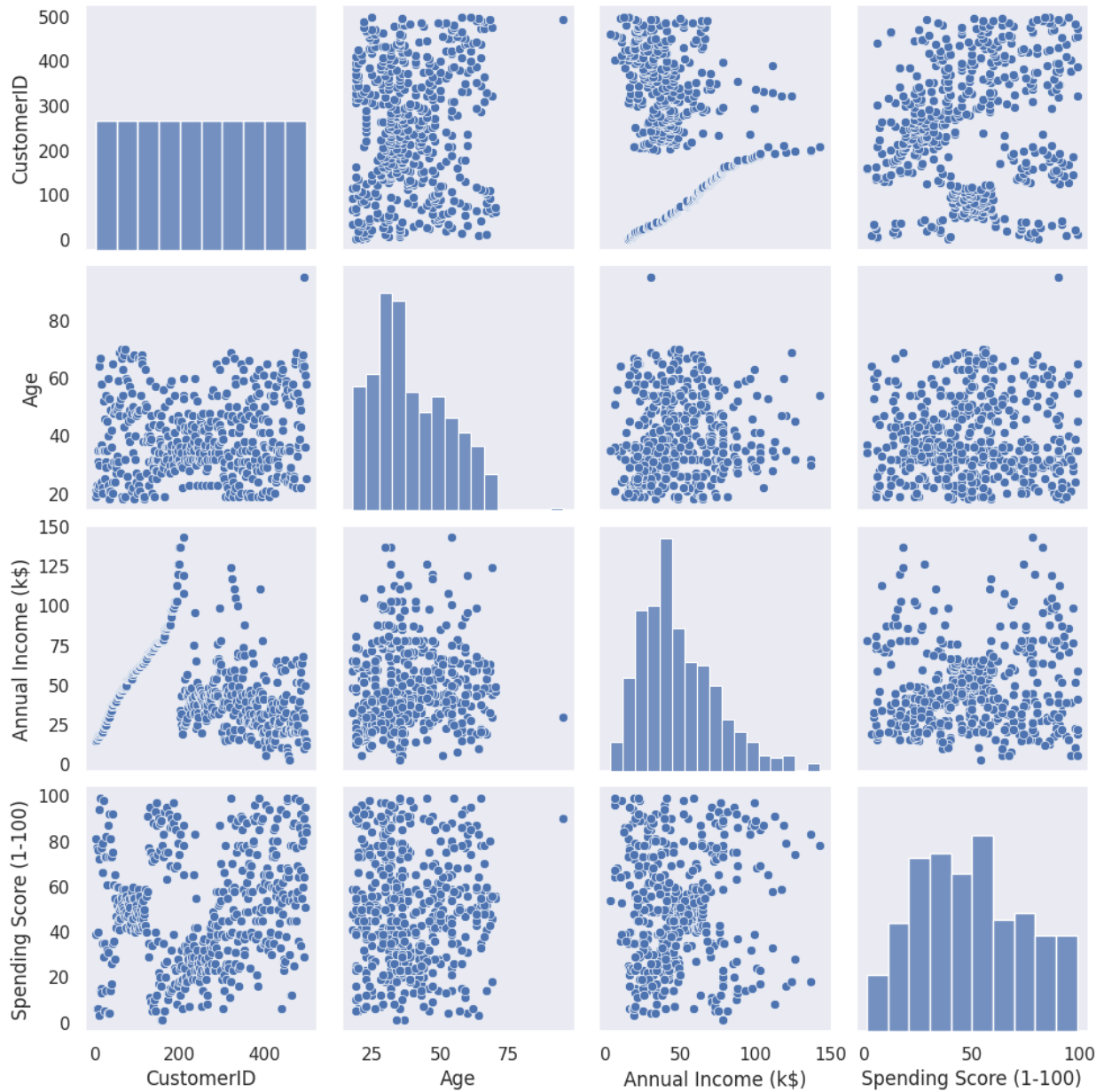
```
# Used to plot- bar plot which counts the occurrences of each unique value
in 'Annual Income(k$)' column
df['Annual Income (k$)'].value_counts().plot.bar(figsize = (13, 6))
plt.title("BAR PLOT FOR ANNUAL INCOME")
plt.xlabel("Unique value of Annual Income")
plt.ylabel("Count")
plt.show()
```



```
# Used to plot- bar plot which counts the occurrences of each unique value
in 'Spending Score(1-100)' column
df['Spending Score (1-100)'].value_counts().plot.bar(figsize = (15, 5))
plt.title("BAR PLOT FOR SPENDING SCORE")
plt.xlabel("Unique value of spending score")
plt.ylabel("Count")
plt.show()
```



```
#pairwise scatter plot matrix of all the numerical variables in df
#This type of visualization is useful for quickly exploring the
relationships between multiple variables in a dataset and identifying any
patterns or trends.
sns.pairplot(df)
```

To create a matrix plot of the correlation matrix for all the numerical variables in df.

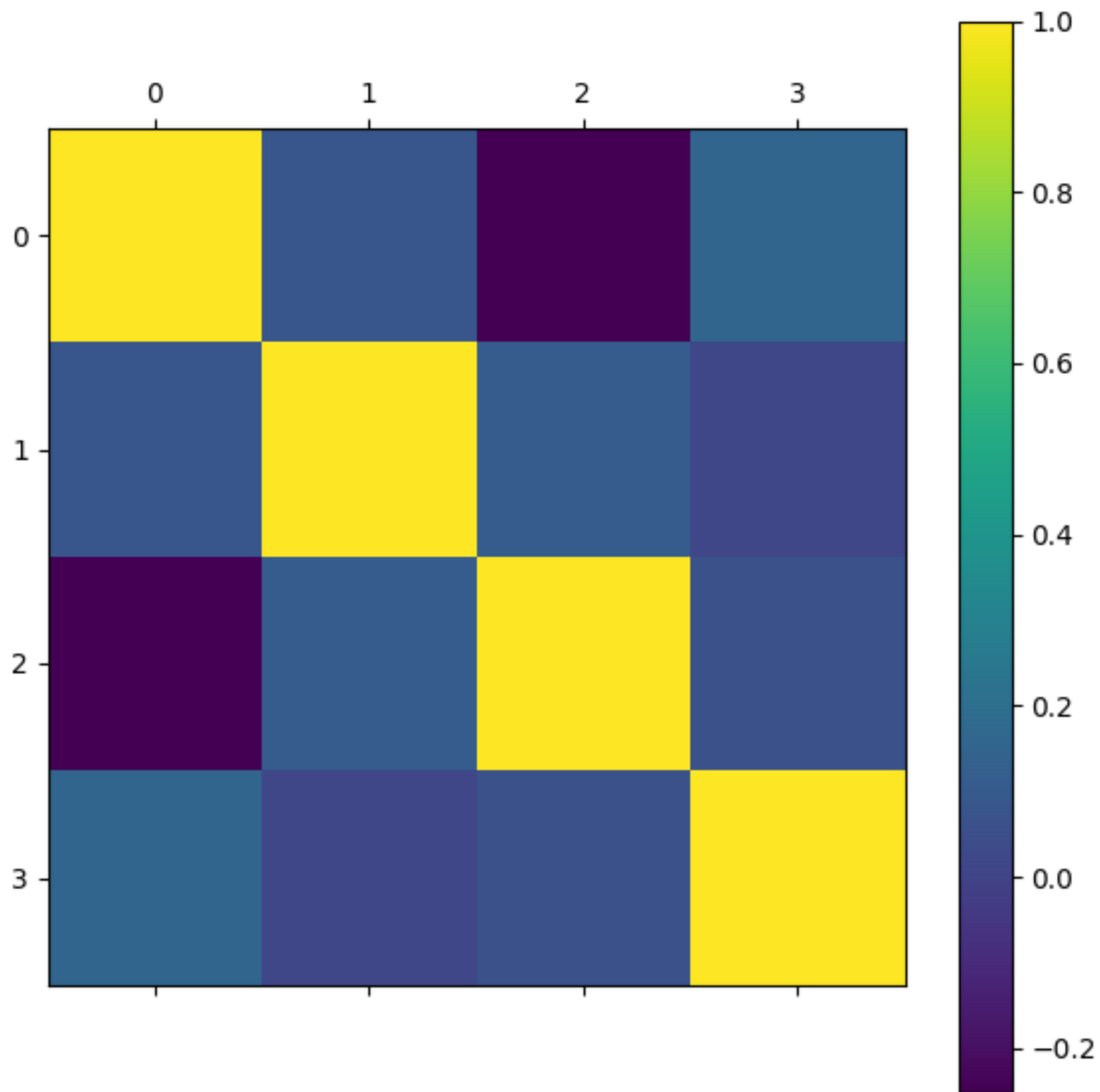
```
import matplotlib.pyplot as plt
```

```
import pandas as pd
```

```
df=pd.read_csv('/content/Mall_Customers.csv')
```

```
plt.matshow(df.corr()) #The corr() function computes the pairwise correlation between columns of a DataFrame
```

```
plt.colorbar()
```



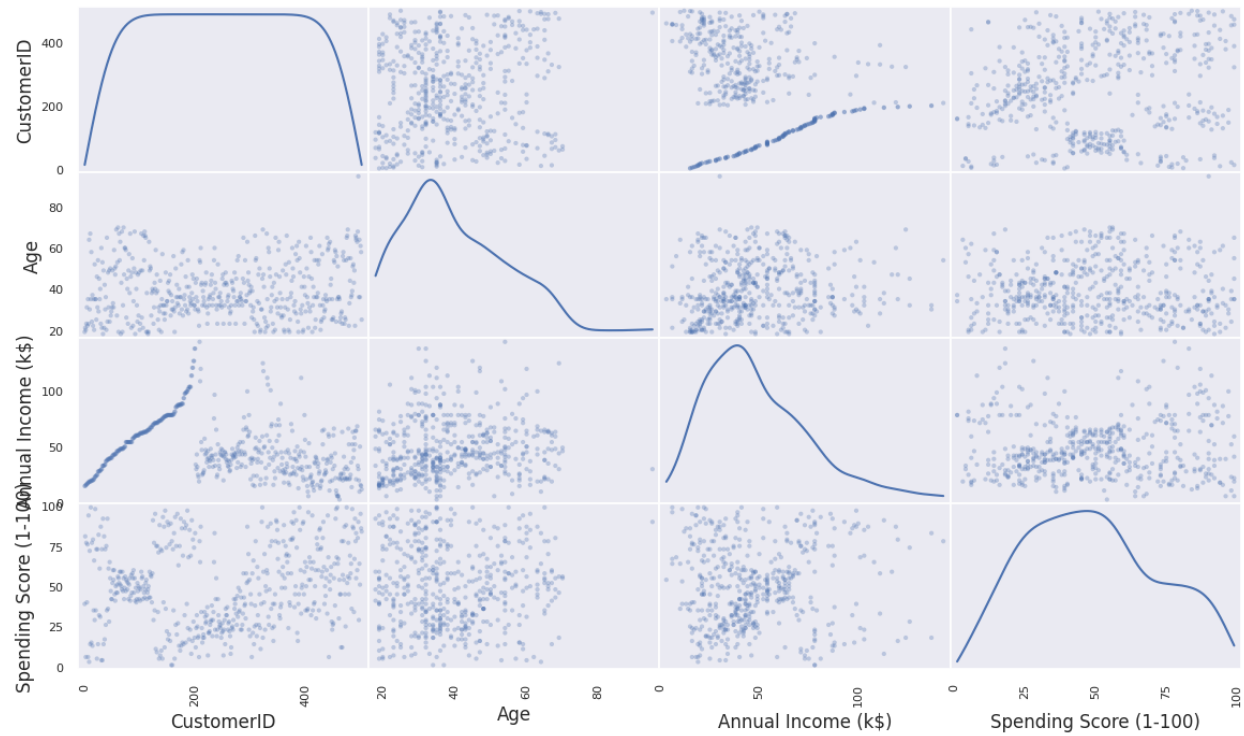
```
# scatter_matrix function from the pandas library to create a scatter plot  
matrix of all the numerical variables in df.
```

```
from pandas.plotting import scatter_matrix
```

```
# alpha sets opacity of scatter plot points, figsize - size of figure will  
be created, diagonal - sets type of diagonal here it is 'kde'
```

```
scatter_matrix(df, alpha = 0.3, figsize = (14,8), diagonal = 'kde')  #kde  
is kernel density estimate
```

```
plt.show()  #alpha=0.3 means opacity is 30%
```

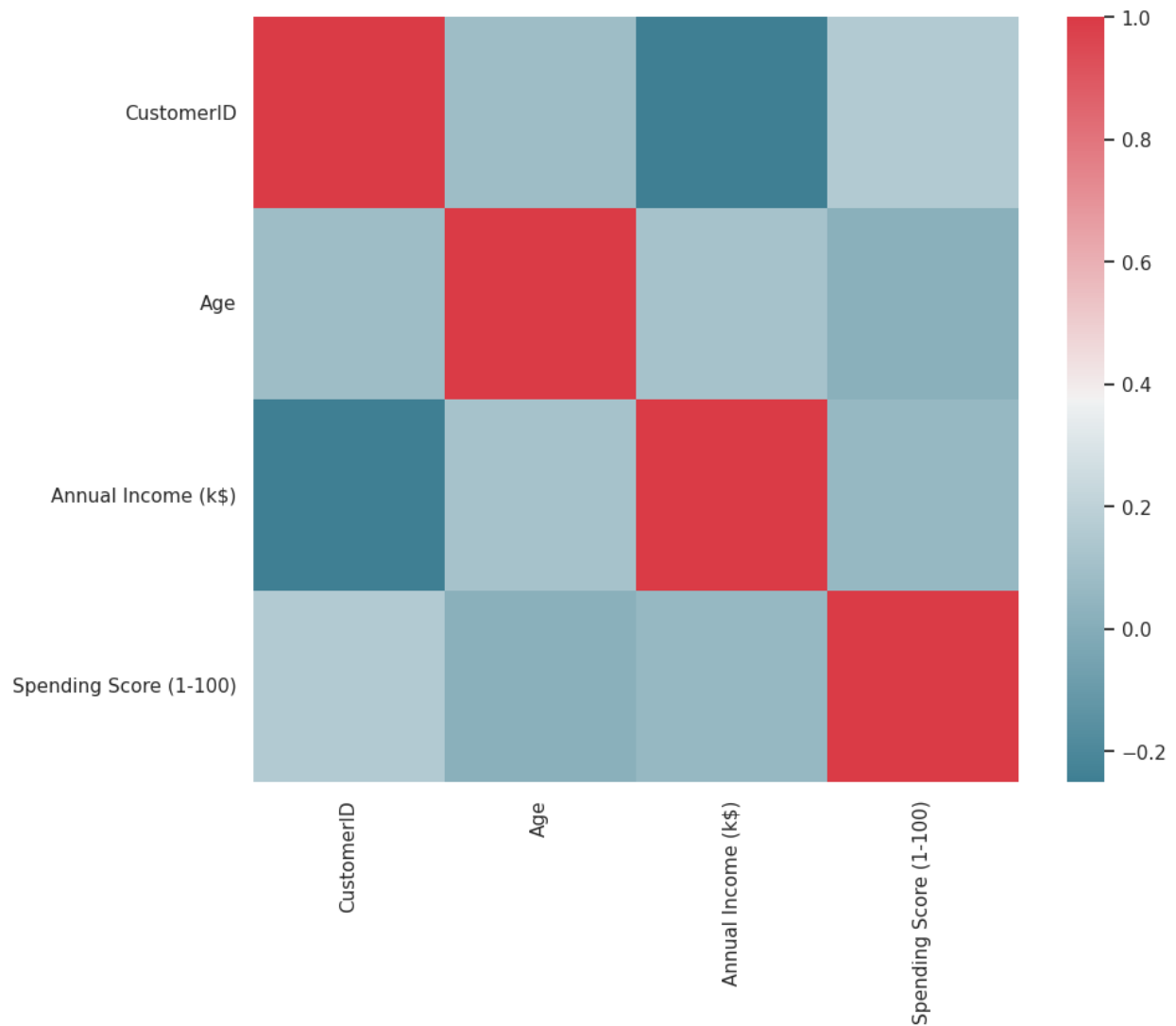


```
# Creating a heatmap
```

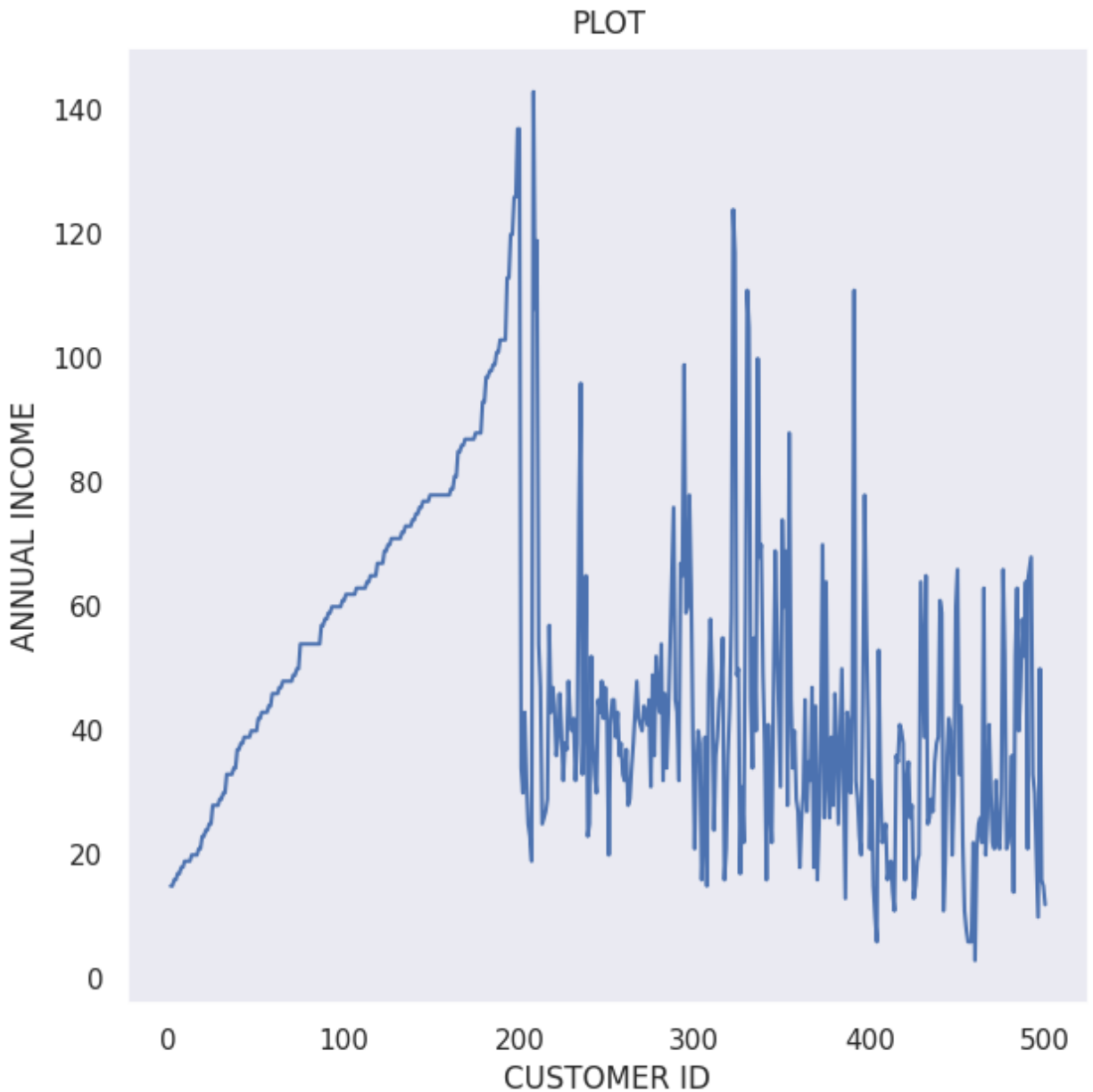
```
fig, axis = plt.subplots(figsize=(10, 8))
```

```
corr = df.corr()
```

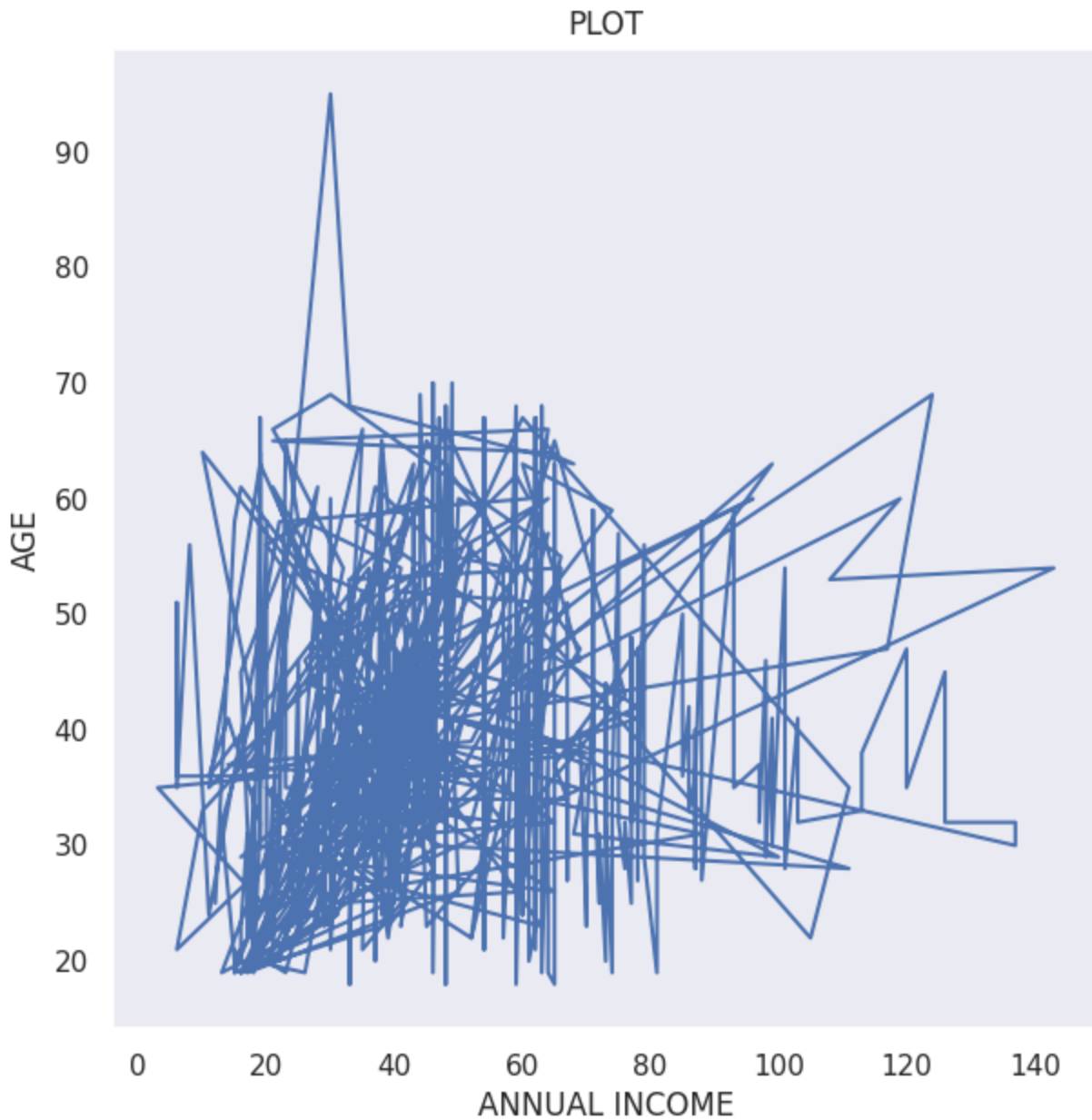
```
# mask used to hide upper-triangular of heat-map, cmap sets colour map,
square sets shape of cells in heatmap as square, ax specifies axis
sns.heatmap(corr, mask = np.zeros_like(corr, dtype = np.bool), cmap =
sns.diverging_palette(220, 10, as_cmap = True),
              square = True, ax = axis)
```



```
# x and y variables are assigned to the 'CustomerID' and 'Annual Income (k$)' columns of the DataFrame
x = df['CustomerID']
y = df['Annual Income (k$)']
plt.xlabel("CUSTOMER ID")
plt.ylabel("ANNUAL INCOME")
plt.title("PLOT")
plt.plot(x, y)
```



```
# x and y variables are assigned to the 'Annual Income(k$)' and 'Age'
columns of the DataFrame
x = df['Annual Income (k$)']
y = df['Age']      #overlapping data points make it difficult to distinguish
therefore jumbled plot.
plt.xlabel("ANNUAL INCOME")
plt.ylabel("AGE")
plt.title("PLOT")
plt.plot(x, y)
```



```
# Enter value of k(number of clusters)
k=int(input("Enter number of clusters: "))
while k==1:
    print("Invalid value of K")
    k=int(input("Enter new value of clusters: "))
kmeans = KMeans(n_clusters=k)
kmeans.fit(x1)
Enter number of clusters: 3
```

/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

```
KMeans(n_clusters=3)
```

K-means clustering

```
identified_clusters = kmeans.fit_predict(x1) #fit_predict() fits the model
to the data and returns the cluster labels for each data point.
```

```
#centroids represent the mean position of the data points within their
respective clusters
```

```
centroids = kmeans.cluster_centers_ #centroids of each cluster are
retrieved
```

```
data_with_clusters = df.copy() #copy of original dataframe
```

```
data_with_clusters['clusters'] = identified_clusters
```

```
print("data with clusters",data_with_clusters)
```

```
plt.scatter(data_with_clusters['Annual Income (k$)'] ,
data_with_clusters['Spending Score
(1-100)'],c=data_with_clusters['clusters'],cmap='viridis',label="luster",m
arker="*")
```

```
plt.scatter(centroids[:,0],centroids[:,1],c='blue',s=100,alpha=0.9)
#creates a scatter plot of the cluster centroids on the same plot as the
data points
```

```
plt.title("Plot with clusters")
```

```
plt.xlabel("ANNUAL INCOME")
```

```
plt.ylabel("SPENDING SCORE")
```

```
plt.legend()
```

```
plt.show()
```

```
data with clusters   CustomerID Gender Age Annual Income (k$) Spending Score (1-100) \
```

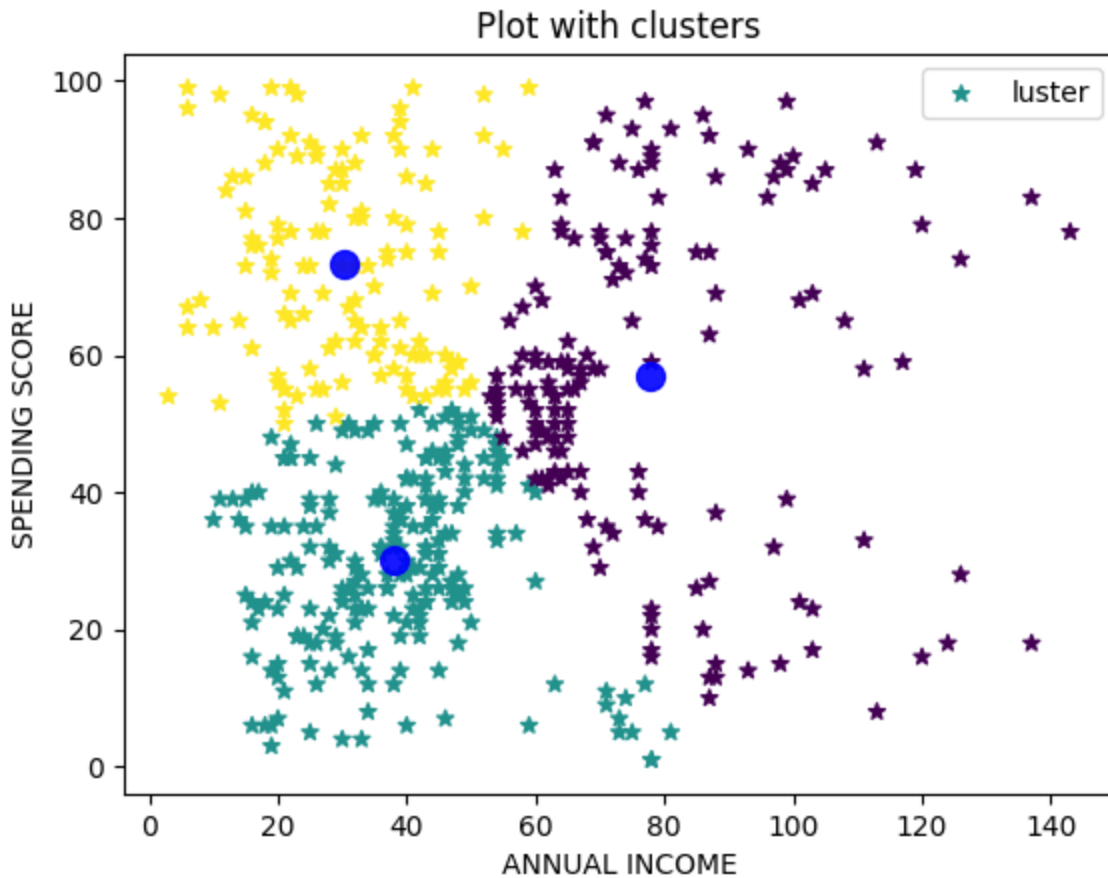
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

..
495	496	Male	64	10	64
496	497	Female	36	50	51
497	498	Male	61	16	95
498	499	Female	58	15	86
499	500	Male	25	12	84

clusters

0	1
1	2
2	1
3	2
4	1
..	...
495	2
496	1
497	2
498	2
499	2

[500 rows x 6 columns]



Elbow Method

In this method, a curve is drawn between “within the sum of squares” (WSS) and the number of clusters.

```
#Importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import seaborn as sns

# load the dataset
```

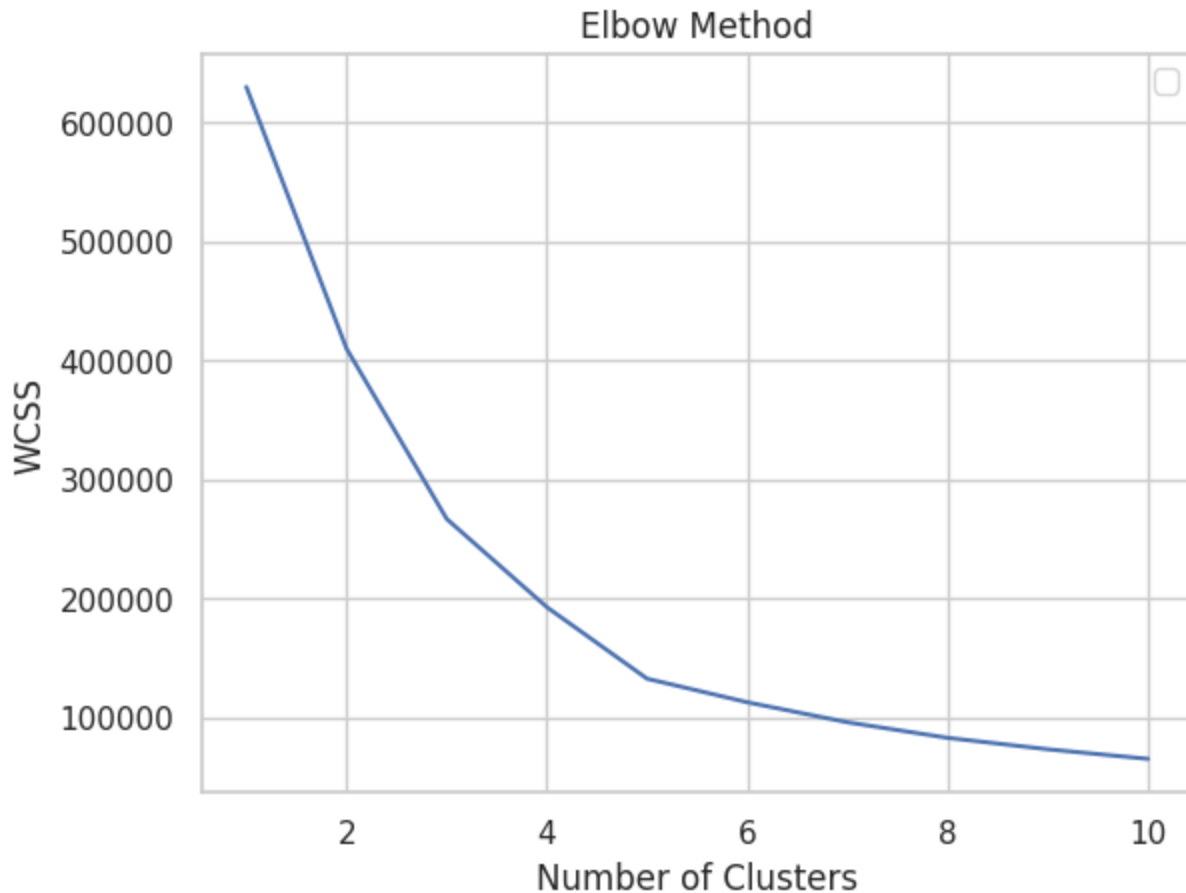
```
df = pd.read_csv('Mall_Customers.csv')

# extract the relevant features for clustering (i.e. Annual Income and
Spending Score)
X = df.iloc[:, [3, 4]].values

# determine the optimal number of clusters using the elbow method
# calculates the Within-Cluster-Sum-of-Squares (WCSS) for different
numbers of clusters, and plots the results to create an elbow curve.
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_) #.inertia() attribute used to retrieve
the WCSS value for that particular number of clusters

    # represents the sum of squared distances of samples to their closest
cluster center.

# plot the elbow curve
sns.set(style = 'whitegrid')
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```



MEAN Shift clustering (It is an iterative clustering technique that does not require any prior information about the number of clusters in the dataset.)

```
from sklearn.cluster import MeanShift, estimate_bandwidth
import numpy as np

# estimate the bandwidth (bandwidth is a parameter for MeanShift)
bandwidth = estimate_bandwidth(X, quantile=0.2, n_samples=len(X))

# The quantile parameter is used to set the fraction of the dataset to be
included in the kernel density estimation

#n_samples parameter sets the number of samples to be used for the
estimation.

# apply MeanShift clustering
```

```

ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)# bin_seeding
parameter is set to True to speed up the clustering process.

ms.fit(X) # Fitting the MeanShift clustering algorithm to the dataset X.


# extract the labels and cluster centers

labels = ms.labels_ # Extracting the cluster labels for each data point
in the dataset.

# Extracting the coordinates of the cluster centers.
cluster_centers = ms.cluster_centers_


# determine the number of clusters
n_clusters_ = len(np.unique(labels))


# print the number of clusters
print("Number of estimated clusters:", n_clusters_)

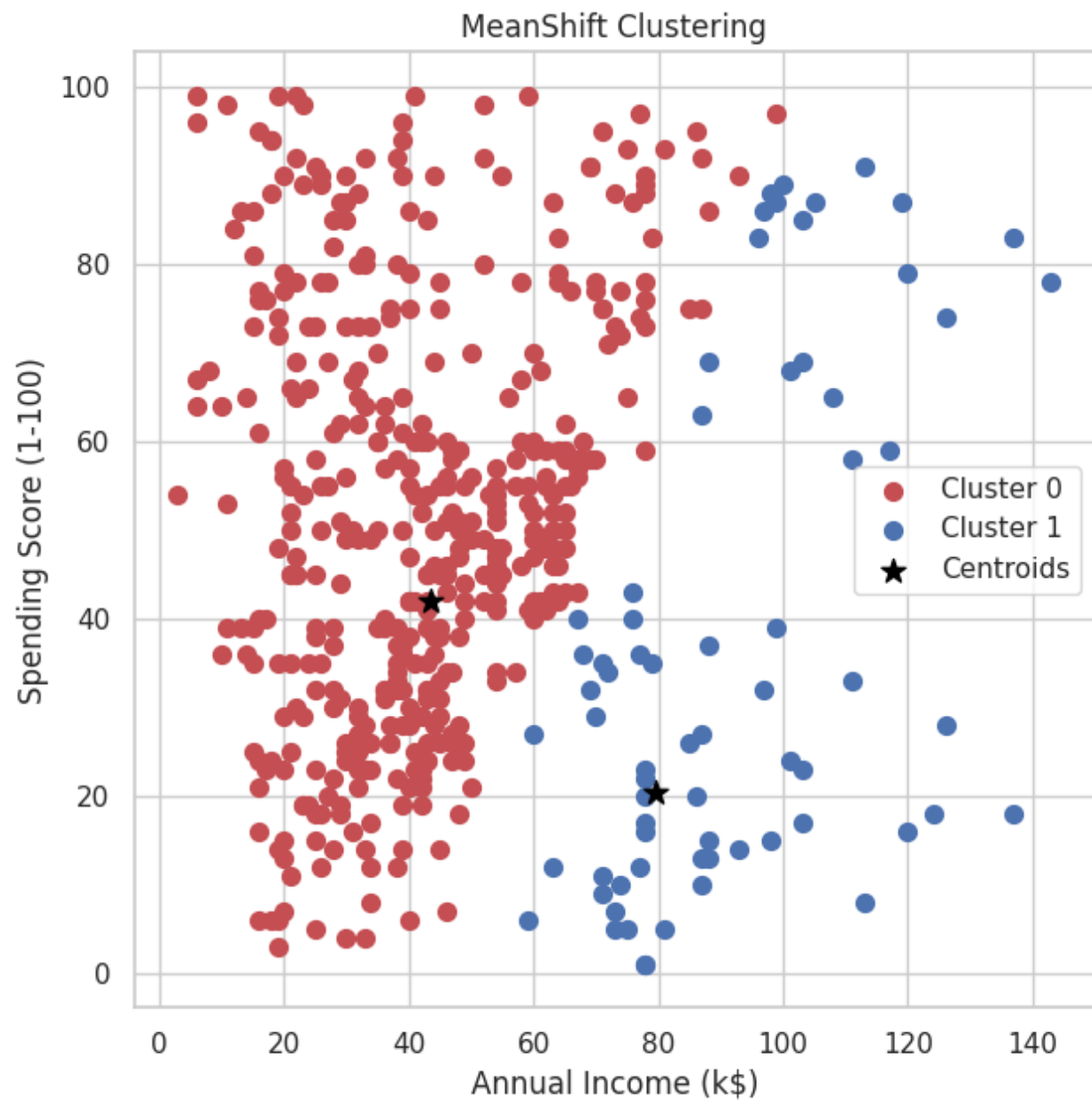

# plot the resulting clusters
colors = ['r', 'b', 'g', 'y', 'c', 'm']
for i in range(n_clusters_):
    plt.scatter(X[labels == i, 0], X[labels == i, 1], s=50, c=colors[i],
label='Cluster %d' % i)


plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], s=100,
c='black', label='Centroids',marker="*")

plt.title('MeanShift Clustering')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()

```

Number of estimated clusters: 2



HIERARCHICAL CLUSTERING;

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import AgglomerativeClustering
```

```
from sklearn.preprocessing import StandardScaler
from scipy.spatial.distance import pdist
from scipy.cluster.hierarchy import dendrogram, linkage

#Agglomerative clustering iteratively merges the two nearest clusters or
data points based on some distance metric until all points or clusters are
in a single cluster.

# Load the dataset
data = pd.read_csv("Mall_Customers.csv")

# Select features for clustering
X = data.iloc[:,3:5].values      # Considering annual income and spending
score

# Standardize the data
scaler = StandardScaler()      #mean=0 and variance=1
X = scaler.fit_transform(X)

# Compute pairwise Euclidean distances
distances = pdist(X, metric='euclidean')      #pdist calculates pairwise
distances between the observations.

# Plot dendrogram to determine optimal number of clusters
Z = linkage(distances, method='ward')      #The linkage function is used to
determine which clusters should be merged at each step of the algorithm
based on a linkage criterion.

dendrogram(Z)

plt.title("Dendrogram")
plt.xlabel("Customers")
plt.ylabel("Euclidean distances")
plt.show()
```

```
# Apply hierarchical clustering

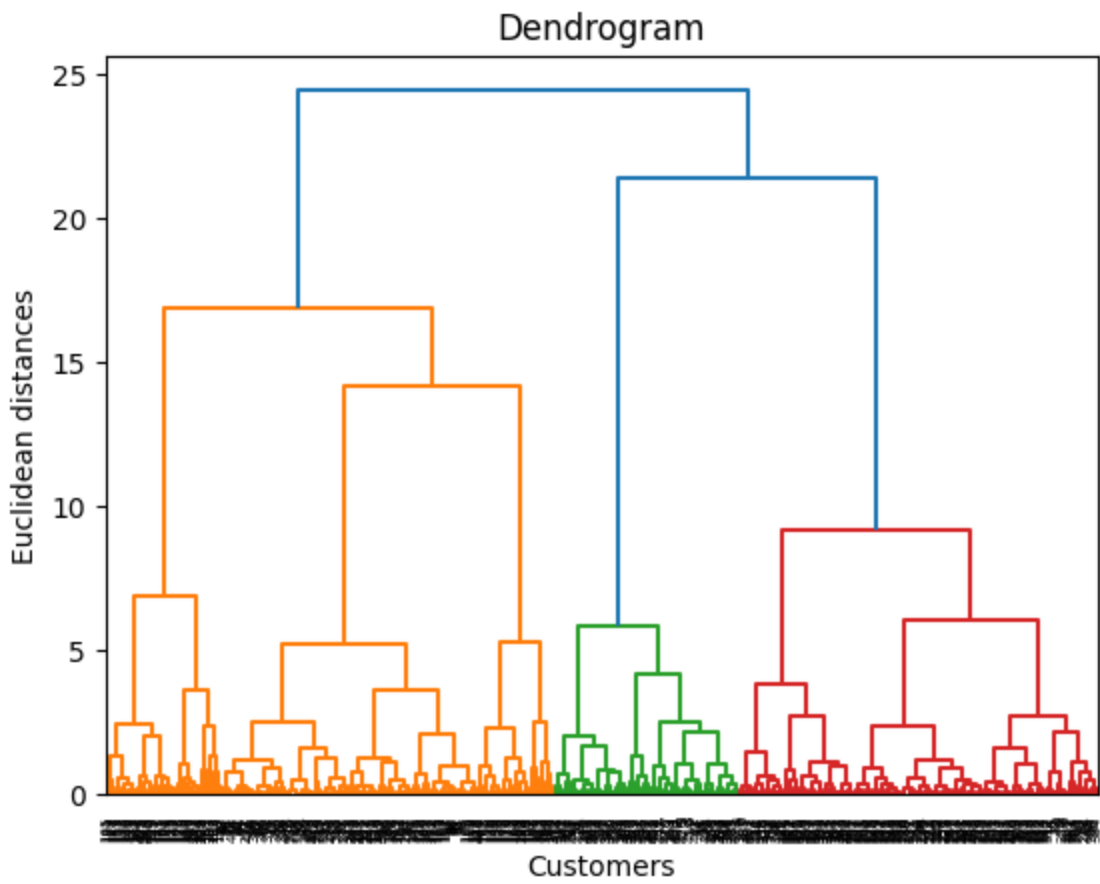
hc = AgglomerativeClustering(n_clusters=5, affinity='euclidean',
                             linkage='ward')      #The ward algorithm is a hierarchical clustering
algorithm that seeks to minimize the sum of squared differences within all
clusters.

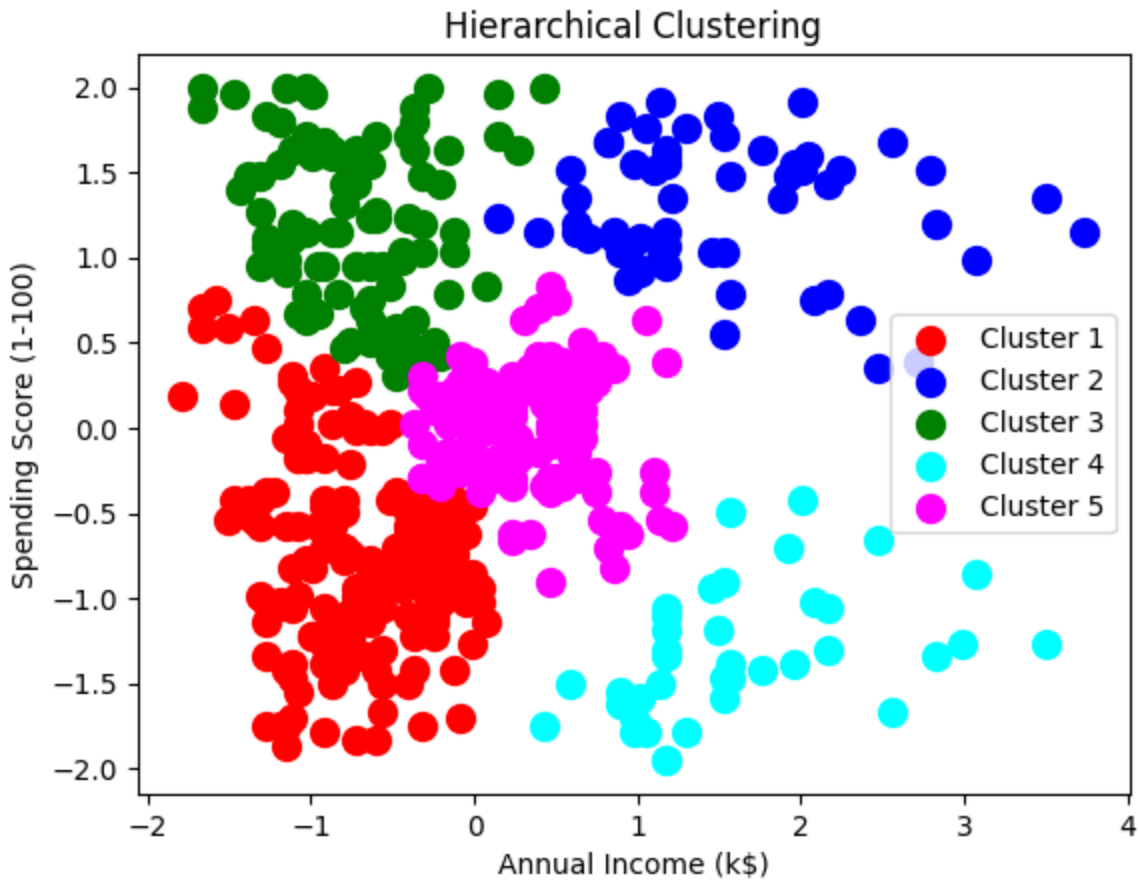
y_hc = hc.fit_predict(X)    # this method will fit and perform predictions
over training data


# Visualize the clusters

plt.scatter(X[y_hc==0,0], X[y_hc==0,1], s=100, c='red', label='Cluster 1')
plt.scatter(X[y_hc==1,0], X[y_hc==1,1], s=100, c='blue', label='Cluster
2')
plt.scatter(X[y_hc==2,0], X[y_hc==2,1], s=100, c='green', label='Cluster
3')
plt.scatter(X[y_hc==3,0], X[y_hc==3,1], s=100, c='cyan', label='Cluster
4')
plt.scatter(X[y_hc==4,0], X[y_hc==4,1], s=100, c='magenta', label='Cluster
5')

plt.title("Hierarchical Clustering")
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.legend()
plt.show()
```





SILHOUETTE METHOD (For comparison of 3 clustering methods)

```
import numpy as np

import pandas as pd

from sklearn.cluster import KMeans, MeanShift, estimate_bandwidth #used
to estimate the bandwidth of a Gaussian kernel for kernel density
estimation (KDE) or mean shift clustering.

from sklearn.metrics import silhouette_score

# load the dataset
df = pd.read_csv('Mall_Customers.csv')

# extract the relevant features for clustering (i.e. Annual Income and
Spending Score)
```

```

X = df.iloc[:, [3, 4]].values

# apply KMeans clustering
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=42)
#init='k-means++' is a commonly used initialization method in K-Means
clustering that can help improve the quality and stability of the
clustering results.

kmeans.fit(X)    #random_state ensures that the algorithm generates the
same sequence of random numbers each time it is run

# apply MeanShift clustering
bandwidth = estimate_bandwidth(X, quantile=0.2, n_samples=len(X))    #
setting quantile=0.2 in Pandas' quantile() function will return the value
that is greater than or equal to 20% of the values in the dataset, which
can be useful for identifying and removing lower outliers.

ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)
ms.fit(X)

# Apply hierarchical clustering
hc = AgglomerativeClustering(n_clusters=5, affinity='euclidean',
linkage='ward') #affinity is a measure of how similar or dissimilar two
data points are.

hc.fit(X)    #Ward linkage is a commonly used linkage criterion in
hierarchical clustering that seeks to minimize the variance within each
cluster.

# evaluate the performance of KMeans and MeanShift using Silhouette score
kmeans_score = silhouette_score(X, kmeans.labels_)
ms_score = silhouette_score(X, ms.labels_)
hc_score=silhouette_score(X,hc.labels_)

#Silhouette score is a metric used to evaluate the quality of clustering
in unsupervised machine learning.

# print the Silhouette score for each algorithm

```

```
print('KMeans Silhouette score: %.3f' % kmeans_score)
print('MeanShift Silhouette score: %.3f' % ms_score)
print('Hierarchical Silhouette score: %.3f' % hc_score)

KMeans Silhouette score: 0.435

MeanShift Silhouette score: 0.402

Hierarchical Silhouette score: 0.392
```

Creation of dashboard

```
# Import necessary libraries
import pandas as pd
import numpy as np

from bokeh.io import output_notebook, show
from bokeh.plotting import figure
from bokeh.models import ColumnDataSource
from bokeh.layouts import row, column
from bokeh.models.widgets import Select, TextInput

# Set the output to be displayed inline in the notebook
output_notebook()

# Create a ColumnDataSource object for the data
source = ColumnDataSource(df)

# Create the first plot
p1 = figure(title='Age vs. Spending Score', plot_width=500,
            plot_height=400)
p1.circle('Age', 'Spending Score (1-100)', source=source)

# Create the second plot
```

```

p2 = figure(title='Annual Income distribution', plot_width=500,
plot_height=400)

hist, edges = np.histogram(df['Annual Income (k$)'], bins=20)

p2.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:],
fill_color='navy', line_color='white')

p2.y_range.start = 0

# Create the dropdown menu widget

menu = Select(options=['Age', 'Annual Income (k$)', 'Spending Score
(1-100)'], value='Age', title='X-axis')

# Create the text input widget

text = TextInput(value='10', title='Number of bins')

# Define the callback function for the widgets

def update():
    x_name = menu.value
    num_bins = int(text.value)

    # Update the first plot
    p1.xaxis.axis_label = x_name
    p1.title.text = f'{x_name} vs. Spending Score'

    # Update the second plot
    hist, edges = np.histogram(df[x_name], bins=num_bins)
    p2.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:],
fill_color='navy', line_color='white')

# Add the widgets and plots to the dashboard

menu.on_change('value', lambda attr, old, new: update())

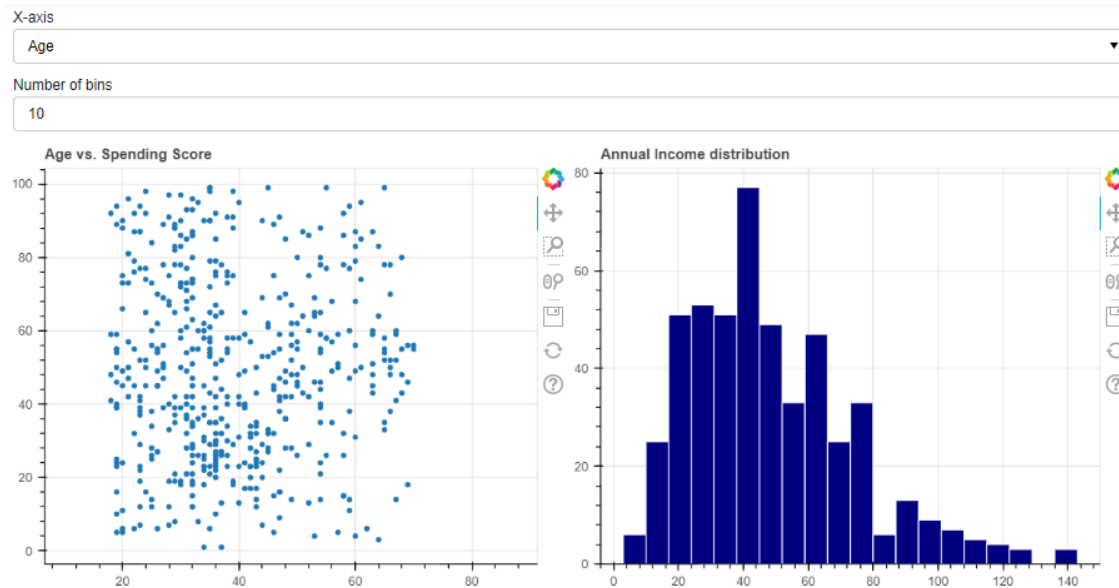
```

```
text.on_change('value', lambda attr, old, new: update())
```

```
layout = column(menu, text, row(p1, p2))
```

```
# Display the dashboard
```

```
show(layout)
```



Sample code for creating dashboard of dendrogram

```
# Import the required libraries

import pandas as pd

import numpy as np

from scipy.cluster.hierarchy import dendrogram, linkage

from bokeh.io import output_notebook, show

from bokeh.models import ColumnDataSource

from bokeh.plotting import figure

from bokeh.layouts import column, row

from bokeh.models.widgets import Select

# Load the dataset
```

```

#df =
pd.read_csv('https://raw.githubusercontent.com/kaustubholpadkar/Datasets/main/Mall_Customers.csv')

# Calculate the linkage matrix using Ward's method
Z = linkage(df.iloc[:, 3:], method='ward')

# Set the output to be displayed inline in the notebook
output_notebook()

# Create a ColumnDataSource object for the dendrogram data
source = ColumnDataSource(pd.DataFrame(dendrogram(Z)['icoord'],
columns=['x0', 'x1', 'x2', 'x3']))

# Create the plot
p = figure(plot_width=800, plot_height=600, tools='pan, wheel_zoom,
reset', title='Dendrogram')

# Add the dendrogram lines to the plot
# Add the dendrogram lines to the plot
dcoord = dendrogram(Z)['dcoord']
ys = []
for i in range(3):
    ys.append(dcoord[:, i])

p.multi_line(xs=[source.data['x0'], source.data['x1'], source.data['x2'],
source.data['x3']], ys=ys, line_color='black', line_width=2)

# Create the dropdown menu widget
menu = Select(options=['ward', 'single', 'complete', 'average'],
value='ward', title='Linkage Method')

```

```

# Define the callback function for the widget
def update(attrname, old, new):
    method = menu.value

    # Update the linkage matrix
    Z = linkage(df.iloc[:, 3:], method=method)

    # Update the dendrogram data source
    source.data = pd.DataFrame(dendrogram(Z)['icoord'], columns=['x0',
'x1', 'x2', 'x3'])

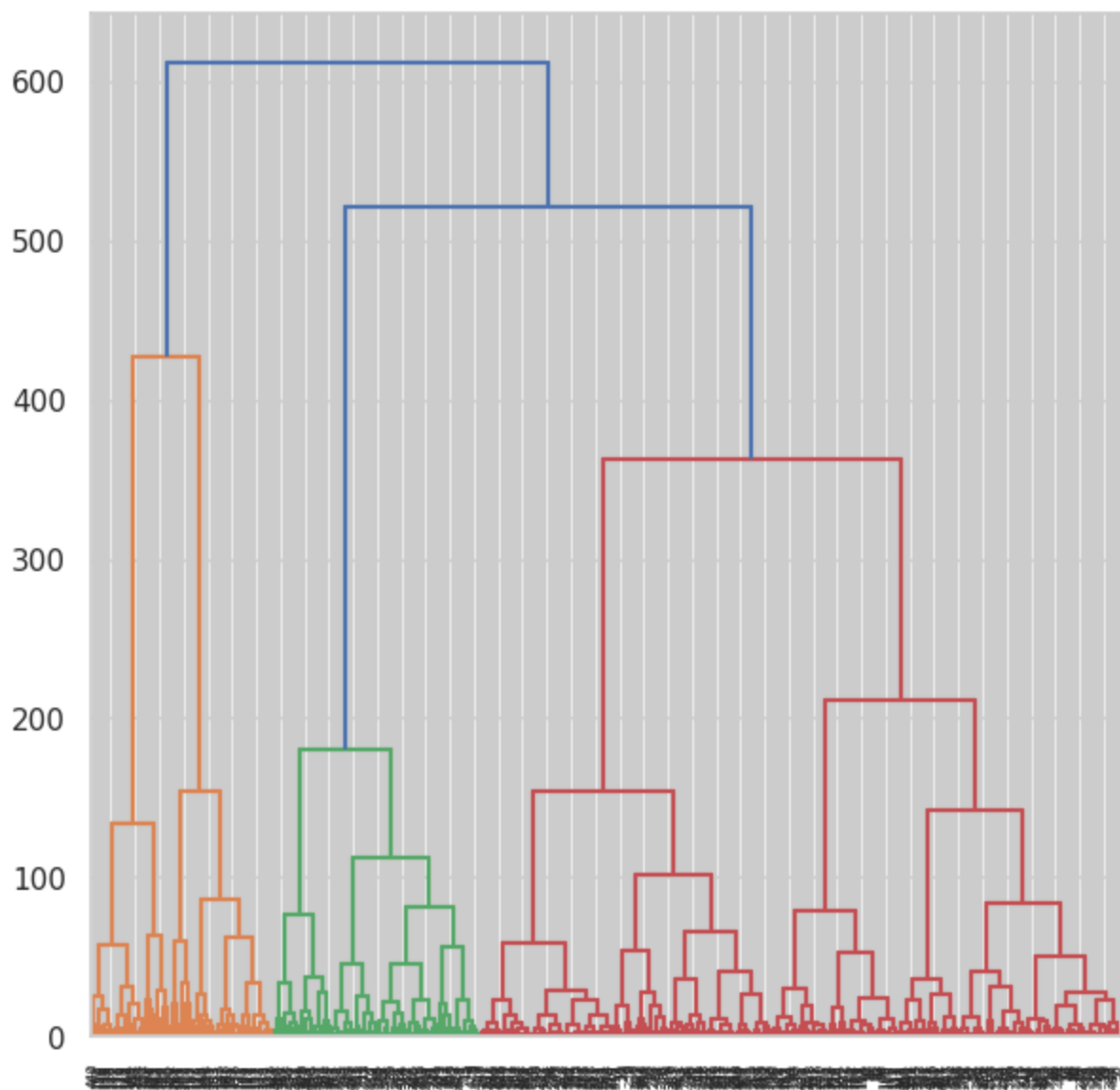
    # Update the plot title
    p.title.text = f'Dendrogram ({method} linkage)'

# Call the update function once to initialize the plot
update(None, None, None)

# Add the widget and plot to the dashboard
menu.on_change('value', update)
layout = column(menu, p)

# Display the dashboard
show(layout)

```



CONCLUSION

The analysis has identified different customer segments based on factors such as demographics, spending habits, purchasing behavior etc. The conclusion of the analysis is recommendations for the mall or retailers within the mall on how to better cater to the needs and preferences of different customer segments. For example, the analysis has found that one segment of customers is more price-sensitive and prefers discounts and promotions, while another segment values high-quality products and is willing to pay a premium price. Based on these findings, the mall or retailers can tailor their marketing and merchandising strategies to appeal to different customer segments and ultimately increase sales and customer satisfaction.

FUTURE SCOPE

Possible area for exploration/ future scope;

1. **Feature engineering:** In this implementation, we have used only two features (Annual Income and Spending Score) to perform customer segmentation. However, there are several other features in the dataset that can be explored to identify different segments of customers.
2. **Visualization techniques:** In the current implementation, we have used scatterplots and pair plots to visualize the clusters. However, there are several other visualization techniques such as t-SNE, PCA, etc. that can be explored to gain a better understanding of the clusters.
3. **Association rule mining:** Association rule mining can be applied to the customer data to identify patterns in customer behavior, such as which products are frequently purchased together or which customers are likely to purchase a particular product.
4. **Predictive analytics:** Machine learning algorithms such as decision trees, random forests, etc. can be applied to the customer data to predict customer behavior, such as which customers are likely to make a purchase or which customers are likely to churn.
5. **Customer feedback analysis:** Customer feedback data can be analyzed to gain insights into customer preferences, satisfaction levels, etc. This can help businesses improve their products and services and provide a better customer experience.

Overall, there are several possibilities for future exploration in mall customer segmentation, and businesses can benefit from implementing these techniques to gain a better understanding of their customers and improve their operations.

REFERENCES

Books

- 1) Machine Learning (Authors: S. Sridhar, M. Vijayalakshmi)
- 2) Artificial Intelligence and Machine Learning (Author: Vinod Chandra)

Online Resources:

- 1) <https://www.kaggle.com/datasets>
- 2) <https://www.analyticsvidhya.com/blog/2021/11/understanding-k-means-clustering-in-machine-learning-with-examples/>