

Customer Segmentation and Analysis

Steps to solve the problem :

1. Importing Libraries.
2. Exploration of data.
3. Data Visualization.
4. Clustering using K-Means.
5. Selection of Clusters.
6. Plotting the Cluster Boundry and Clusters.
7. 3D Plot of Clusters.

Importing Libraries.

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
import plotly as py
import plotly.graph_objs as go
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.metrics import silhouette_score
import scipy.cluster.hierarchy as sch
```

Data Exploration

```
!mkdir ~/.kaggle
```

```
↳ mkdir: cannot create directory '/root/.kaggle': File exists
```

```
cp kaggle.json ~/.kaggle
```

```
!chmod 600 /root/.kaggle/kaggle.json
```

```
!kaggle datasets download -d vjchoudhary7/customer-segmentation-tutorial-in-python
```

```
↳ Dataset URL: https://www.kaggle.com/datasets/vjchoudhary7/customer-segmentation-tutorial-in-python
License(s): other
customer-segmentation-tutorial-in-python.zip: Skipping, found more recently modified local copy (use --force to force do
```

```
!unzip -qq customer-segmentation-tutorial-in-python.zip
```

```
↳ replace Mall_Customers.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

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

```
↳
```

	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

```
# df["Gender"] = df["Genre"]
# df.drop(["Genre"], axis=1, inplace=True)
# Check if 'Genre' column exists before renaming
if 'Genre' in df.columns:
    df["Gender"] = df["Genre"]
    df.drop(["Genre"], axis=1, inplace=True)
else:
    print("Column 'Genre' not found in the DataFrame.")
```

Column 'Genre' not found in the DataFrame.

```
df.head()
```

```
df.head()
```

	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

```
df.shape
```

```
(200, 5)
```

```
df.describe()
```

```
df.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

```
df.dtypes
```

```
df.dtypes
```

CustomerID	int64
Gender	object
Age	int64
Annual Income (k\$)	int64
Spending Score (1-100)	int64
dtype:	object

```
df.isnull().sum()
```

```
df.isnull().sum()
```


CustomerID	0
Gender	0
Age	0
Annual Income (k\$)	0
Spending Score (1-100)	0
dtype:	int64

✓ Data Visualization

```
plt.style.use('fivethirtyeight')
```

✓ Histograms

```
plt.figure(1, figsize = (15, 6))
n = 0
for x in ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
    n += 1
    plt.subplot(1, 3, n)
    plt.subplots_adjust(hspace=0.5, wspace=0.5)
    sns.distplot(df[x], bins=20)
    plt.title('Distplot of {}'.format(x))
plt.show()
```

 <ipython-input-26-d33c6fc935e4>:7: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(df[x] , bins = 20)
<ipython-input-26-d33c6fc935e4>:7: UserWarning:
```

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

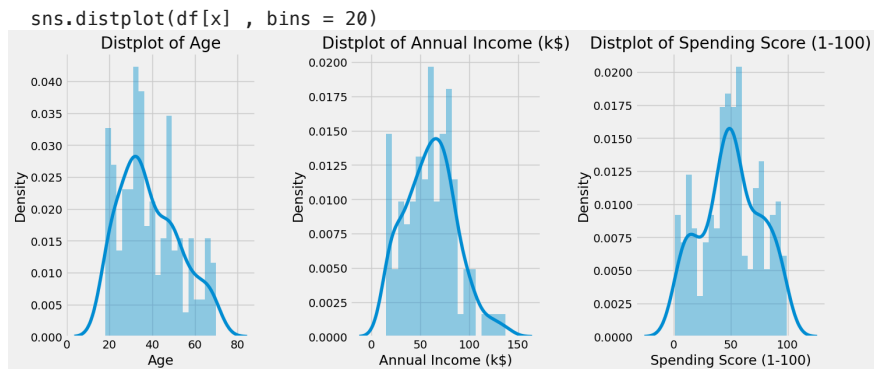
For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>

```
sns.distplot(df[x] , bins = 20)
<ipython-input-26-d33c6fc935e4>:7: UserWarning:
```

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

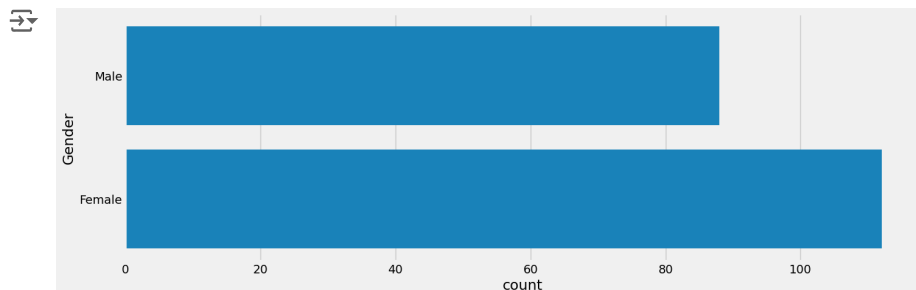
Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see <https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751>



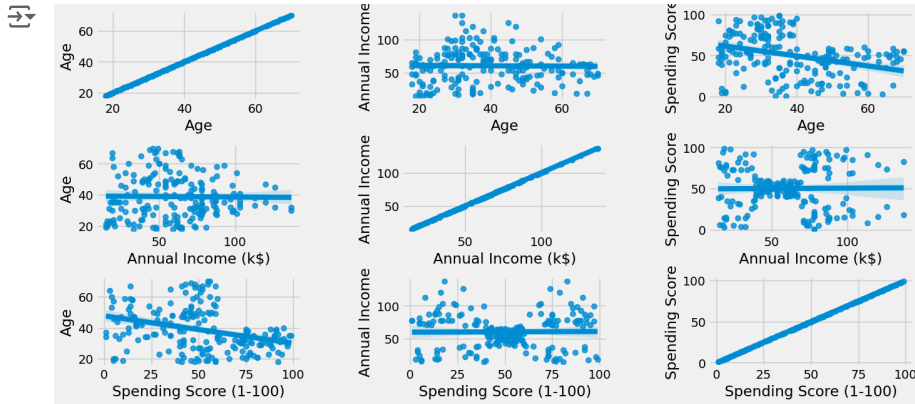
✓ Count Plot of Gender

```
plt.figure(1 , figsize = (15 , 5))
sns.countplot(y = 'Gender' , data = df)
plt.show()
```

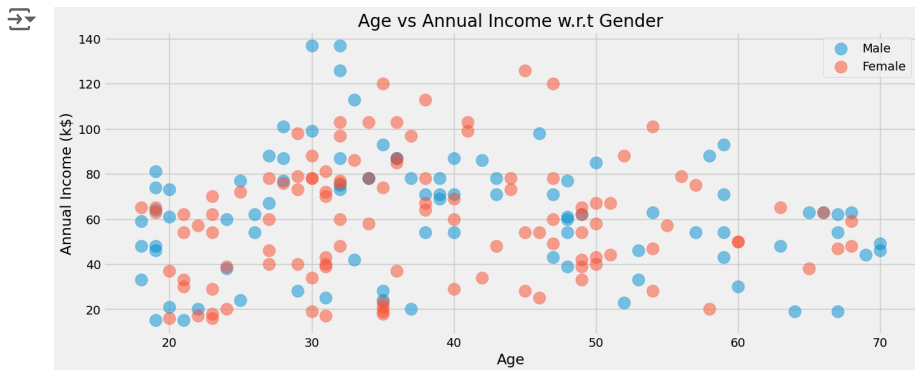


✓ Plotting the Relation between Age , Annual Income and Spending Score

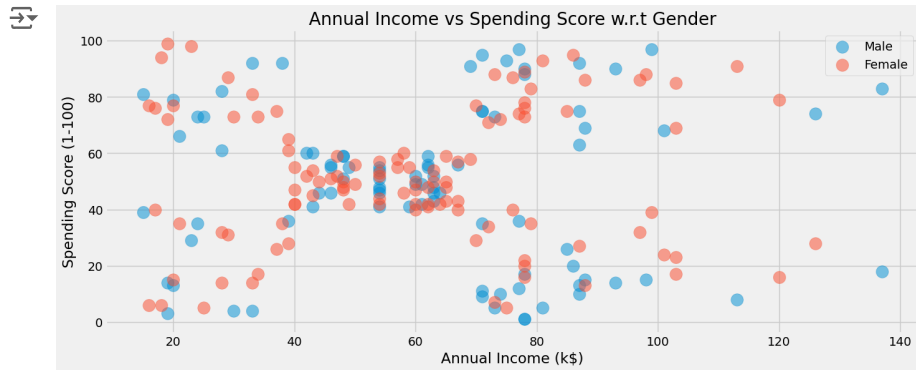
```
plt.figure(1, figsize = (15, 7))
n = 0
for x in ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
    for y in ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
        n += 1
        plt.subplot(3, 3, n)
        plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
        sns.regplot(x = x, y = y, data = df)
        plt.ylabel(y.split()[0]+' '+y.split()[1] if len(y.split()) > 1 else y)
plt.show()
```



```
plt.figure(1, figsize = (15, 6))
for gender in ['Male', 'Female']:
    plt.scatter(x = 'Age', y = 'Annual Income (k$)', data = df[df['Gender'] == gender],
               s = 200, alpha = 0.5, label = gender)
plt.xlabel('Age'), plt.ylabel('Annual Income (k$)')
plt.title('Age vs Annual Income w.r.t Gender')
plt.legend()
plt.show()
```



```
plt.figure(1, figsize = (15, 6))
for gender in ['Male', 'Female']:
    plt.scatter(x = 'Annual Income (k$)', y = 'Spending Score (1-100)',
                data = df[df['Gender'] == gender], s = 200, alpha = 0.5, label = gender)
plt.xlabel('Annual Income (k$)', plt.ylabel('Spending Score (1-100)')
plt.title('Annual Income vs Spending Score w.r.t Gender')
plt.legend()
plt.show()
```



▼ Distribution of values in Age , Annual Income and Spending Score according to Gender

```
plt.figure(1, figsize = (15, 7))
n = 0
for cols in ['Age', 'Annual Income (k$)', 'Spending Score (1-100)']:
    n += 1
    plt.subplot(1, 3, n)
    plt.subplots_adjust(hspace = 0.5, wspace = 0.5)
    sns.violinplot(x = cols, y = 'Gender', data = df, palette = 'vlag')
    sns.swarmplot(x = cols, y = 'Gender', data = df)
    plt.ylabel('Gender' if n == 1 else '')
    plt.title('Boxplots & Swarmplots' if n == 2 else '')
plt.show()
```

```
<ipython-input-30-296c16218033>:7: FutureWarning:
```

Passing `palette` without assigning `hue` is deprecated and will be removed in

```
sns.violinplot(x = cols , y = 'Gender' , data = df , palette = 'vlag')
```

```
<ipython-input-30-296c16218033>:7: FutureWarning:
```

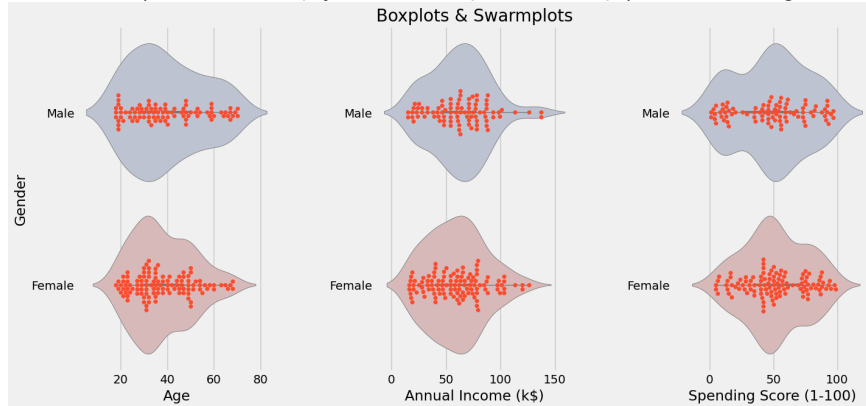
Passing `palette` without assigning `hue` is deprecated and will be removed in

```
sns.violinplot(x = cols , y = 'Gender' , data = df , palette = 'vlag')
```

```
<ipython-input-30-296c16218033>:7: FutureWarning:
```

Passing `palette` without assigning `hue` is deprecated and will be removed in

```
sns.violinplot(x = cols , y = 'Gender' , data = df , palette = 'vlag')
```



✓ Clustering using K- means

1.Segmentation using Age and Spending Score

```
'''Age and spending Score'''
```

```
X1 = df[['Age' , 'Spending Score (1-100)']].iloc[:, :].values
```

```
inertia = []
```

```
for n in range(1 , 11):
```

```
    algorithm = (KMeans(n_clusters = n ,init='k-means++', n_init = 10 ,max_iter=300,  
                        tol=0.0001, random_state= 111 , algorithm='elkan') )
```

```
    algorithm.fit(X1)
```

```
    inertia.append(algorithm.inertia_)
```

```
<ipython-input-30-296c16218033>:13: RuntimeWarning: algorithm='elkan' doesn't make  
warnings.warn()
```

✓ Selecting N Clusters based in Inertia (Squared Distance between Centroids and data points, should be less)

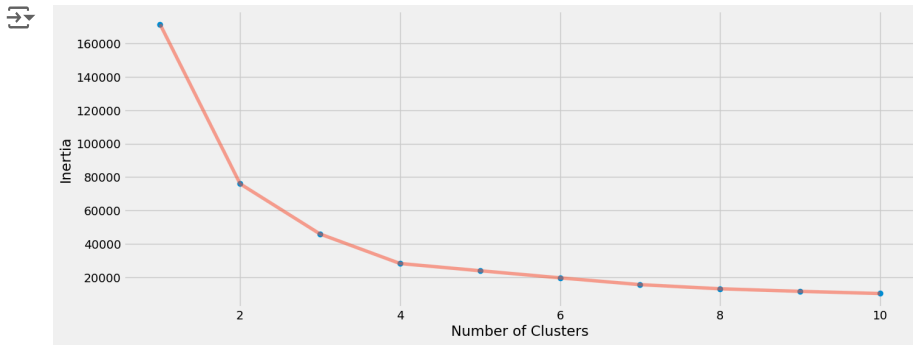
```
plt.figure(1 , figsize = (15 ,6))
```

```
plt.plot(np.arange(1 , 11) , inertia , 'o')
```

```
plt.plot(np.arange(1 , 11) , inertia , '-' , alpha = 0.5)
```

```
plt.xlabel('Number of Clusters') , plt.ylabel('Inertia')
```

```
plt.show()
```



```

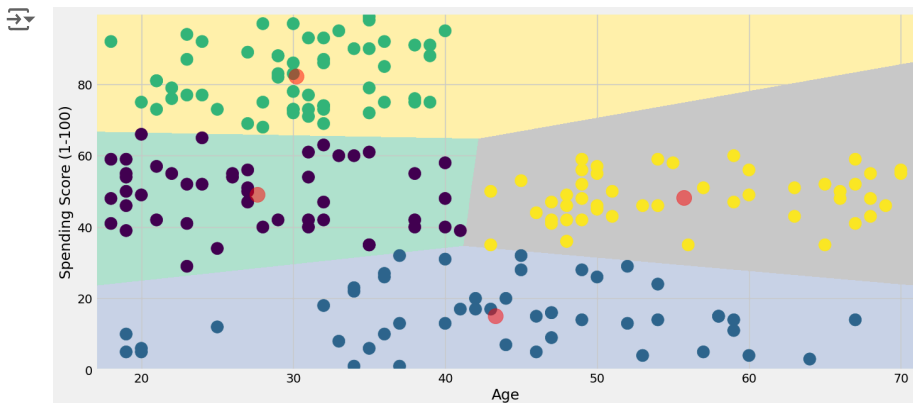
algorithm = (KMeans(n_clusters = 4 ,init='k-means++', n_init = 10 ,max_iter=300,
                    tol=0.0001, random_state= 111 , algorithm='elkan') )
algorithm.fit(X1)
labels1 = algorithm.labels_
centroids1 = algorithm.cluster_centers_

h = 0.02
x_min, x_max = X1[:, 0].min() - 1, X1[:, 0].max() + 1
y_min, y_max = X1[:, 1].min() - 1, X1[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = algorithm.predict(np.c_[xx.ravel(), yy.ravel()])

plt.figure(1 , figsize = (15 , 7) )
plt.clf()
Z = Z.reshape(xx.shape)
plt.imshow(Z , interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap = plt.cm.Pastel2, aspect = 'auto', origin='lower')

plt.scatter( x = 'Age' , y = 'Spending Score (1-100)' , data = df , c = labels1 ,
            s = 200 )
plt.scatter(x = centroids1[:, 0] , y = centroids1[:, 1] , s = 300 , c = 'red' , alpha = 0.5)
plt.ylabel('Spending Score (1-100)' , plt.xlabel('Age')
plt.show()

```



2. Segmentation using Annual Income and Spending Score

```

'''Annual Income and spending Score'''
X2 = df[['Annual Income (k$)', 'Spending Score (1-100)']].iloc[:, :].values
inertia = []
for n in range(1, 11):
    algorithm = (KMeans(n_clusters = n ,init='k-means++', n_init = 10 ,max_iter=300,
                        tol=0.0001, random_state= 111 , algorithm='elkan') )
    algorithm.fit(X2)
    inertia.append(algorithm.inertia_)

```

```

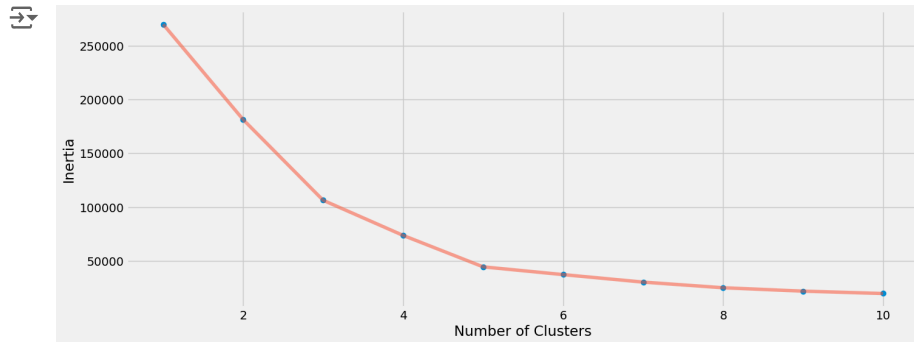
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1373: RuntimeWarning: algorithm='elkan' doesn't make
warnings.warn(

```

```

plt.figure(1 , figsize = (15 ,6))
plt.plot(np.arange(1 , 11) , inertia , 'o')
plt.plot(np.arange(1 , 11) , inertia , '-' , alpha = 0.5)
plt.xlabel('Number of Clusters') , plt.ylabel('Inertia')
plt.show()

```



```

algorithm = (KMeans(n_clusters = 5 ,init='k-means++', n_init = 10 ,max_iter=300,
                    tol=0.0001, random_state= 111 , algorithm='elkan') )
algorithm.fit(X2)
labels2 = algorithm.labels_
centroids2 = algorithm.cluster_centers_

```

```

h = 0.02
x_min, x_max = X2[:, 0].min() - 1, X2[:, 0].max() + 1
y_min, y_max = X2[:, 1].min() - 1, X2[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z2 = algorithm.predict(np.c_[xx.ravel(), yy.ravel()])

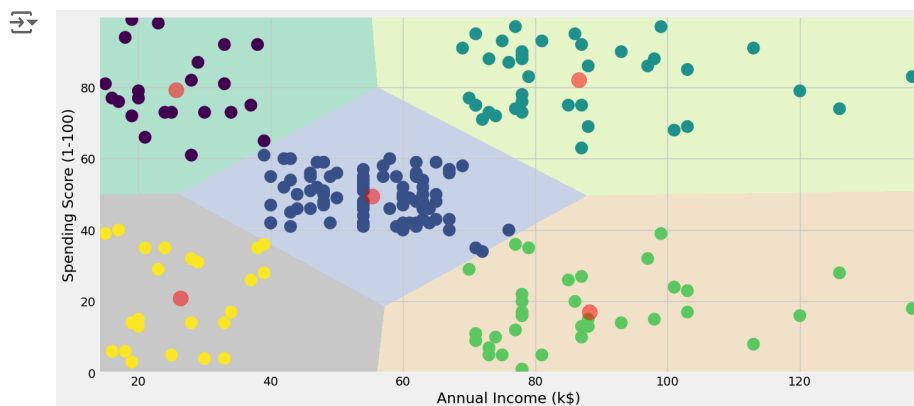
```

```

plt.figure(1 , figsize = (15 , 7) )
plt.clf()
Z2 = Z2.reshape(xx.shape)
plt.imshow(Z2 , interpolation='nearest',
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap = plt.cm.Pastel2, aspect = 'auto', origin='lower')

plt.scatter( x = 'Annual Income (k$)' , y = 'Spending Score (1-100)' , data = df , c = labels2 ,
            s = 200 )
plt.scatter(x = centroids2[:, 0] , y = centroids2[:, 1] , s = 300 , c = 'red' , alpha = 0.5)
plt.ylabel('Spending Score (1-100)') , plt.xlabel('Annual Income (k$)')
plt.show()

```



3.Segmentation using Age , Annual Income and Spending Score


```

X3 = df[['Age' , 'Annual Income (k$)' , 'Spending Score (1-100)']].iloc[:, :].values
inertia = []
for n in range(1 , 11):
    algorithm = (KMeans(n_clusters = n , init='k-means++' , n_init = 10 , max_iter=300,
                        tol=0.0001, random_state= 111 , algorithm='elkan') )
    algorithm.fit(X3)
    inertia.append(algorithm.inertia_)

```

```

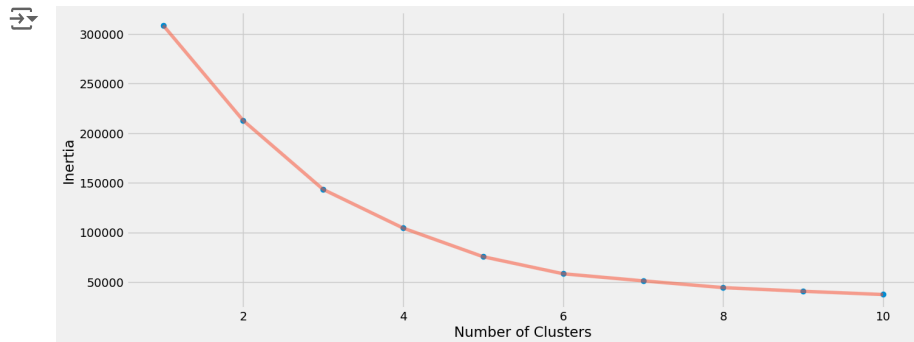
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:1373: RuntimeWarning: algorithm='elkan' doesn't make
warnings.warn(

```

```

plt.figure(1 , figsize = (15 ,6))
plt.plot(np.arange(1 , 11) , inertia , 'o')
plt.plot(np.arange(1 , 11) , inertia , '-' , alpha = 0.5)
plt.xlabel('Number of Clusters') , plt.ylabel('Inertia')
plt.show()

```



```

algorithm = (KMeans(n_clusters = 6 , init='k-means++' , n_init = 10 , max_iter=300,
                    tol=0.0001, random_state= 111 , algorithm='elkan') )
algorithm.fit(X3)
labels3 = algorithm.labels_
centroids3 = algorithm.cluster_centers_

```

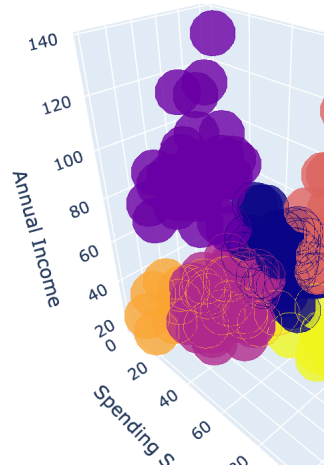
```

df['label3'] = labels3
trace1 = go.Scatter3d(
    x= df['Age'],
    y= df['Spending Score (1-100)'],
    z= df['Annual Income (k$)'],
    mode='markers',
    marker=dict(
        color = df['label3'],
        size= 20,
        line=dict(
            color= df['label3'],
            width= 12
        ),
        opacity=0.8
    )
)
data = [trace1]
layout = go.Layout(
    # margin=dict(
    #     l=0,
    #     r=0,
    #     b=0,
    #     t=0
    # )
    title= 'Clusters',
    scene = dict(
        xaxis = dict(title = 'Age'),
        yaxis = dict(title = 'Spending Score'),
        zaxis = dict(title = 'Annual Income')
    )
)
fig = go.Figure(data=data, layout=layout)
py.offline.iplot(fig)

```



Clusters



✓ DBScan

```
eps_values = [0.01, 0.0001, 0.2, 0.3, 0.4, 0.5, 1.0, 1.5, 2.0] # Adjust as needed
min_samples_values = [1, 2, 3, 4, 5, 10, 15] # Adjust as needed

best_score = -1
best_eps = None
best_min_samples = None

for eps in eps_values:
    for min_samples in min_samples_values:
        dbscan = DBSCAN(eps=eps, min_samples=min_samples)
        labels = dbscan.fit_predict(X3)
        score = silhouette_score(X3, labels3)
        print(f'eps={eps}, min_samples={min_samples}, silhouette_score={score}')

        if score > best_score:
            best_score = score
            best_eps = eps
            best_min_samples = min_samples

print(f'Best Parameters: eps={best_eps}, min_samples={best_min_samples}, silhouette_score={best_score}')
```

```
eps=0.01, min_samples=1, silhouette_score=0.4523443947724053
eps=0.01, min_samples=2, silhouette_score=0.4523443947724053
eps=0.01, min_samples=3, silhouette_score=0.4523443947724053
eps=0.01, min_samples=4, silhouette_score=0.4523443947724053
eps=0.01, min_samples=5, silhouette_score=0.4523443947724053
eps=0.01, min_samples=10, silhouette_score=0.4523443947724053
eps=0.01, min_samples=15, silhouette_score=0.4523443947724053
eps=0.0001, min_samples=1, silhouette_score=0.4523443947724053
eps=0.0001, min_samples=2, silhouette_score=0.4523443947724053
eps=0.0001, min_samples=3, silhouette_score=0.4523443947724053
eps=0.0001, min_samples=4, silhouette_score=0.4523443947724053
eps=0.0001, min_samples=5, silhouette_score=0.4523443947724053
eps=0.0001, min_samples=10, silhouette_score=0.4523443947724053
eps=0.0001, min_samples=15, silhouette_score=0.4523443947724053
eps=0.2, min_samples=1, silhouette_score=0.4523443947724053
eps=0.2, min_samples=2, silhouette_score=0.4523443947724053
eps=0.2, min_samples=3, silhouette_score=0.4523443947724053
eps=0.2, min_samples=4, silhouette_score=0.4523443947724053
eps=0.2, min_samples=5, silhouette_score=0.4523443947724053
eps=0.2, min_samples=10, silhouette_score=0.4523443947724053
eps=0.2, min_samples=15, silhouette_score=0.4523443947724053
eps=0.3, min_samples=1, silhouette_score=0.4523443947724053
eps=0.3, min_samples=2, silhouette_score=0.4523443947724053
eps=0.3, min_samples=3, silhouette_score=0.4523443947724053
eps=0.3, min_samples=4, silhouette_score=0.4523443947724053
eps=0.3, min_samples=5, silhouette_score=0.4523443947724053
eps=0.3, min_samples=10, silhouette_score=0.4523443947724053
eps=0.3, min_samples=15, silhouette_score=0.4523443947724053
eps=0.4, min_samples=1, silhouette_score=0.4523443947724053
```

```

eps=0.4, min_samples=2, silhouette_score=0.4523443947724053
eps=0.4, min_samples=3, silhouette_score=0.4523443947724053
eps=0.4, min_samples=4, silhouette_score=0.4523443947724053
eps=0.4, min_samples=5, silhouette_score=0.4523443947724053
eps=0.4, min_samples=10, silhouette_score=0.4523443947724053
eps=0.4, min_samples=15, silhouette_score=0.4523443947724053
eps=0.5, min_samples=1, silhouette_score=0.4523443947724053
eps=0.5, min_samples=2, silhouette_score=0.4523443947724053
eps=0.5, min_samples=3, silhouette_score=0.4523443947724053
eps=0.5, min_samples=4, silhouette_score=0.4523443947724053
eps=0.5, min_samples=5, silhouette_score=0.4523443947724053
eps=0.5, min_samples=10, silhouette_score=0.4523443947724053
eps=0.5, min_samples=15, silhouette_score=0.4523443947724053
eps=1.0, min_samples=1, silhouette_score=0.4523443947724053
eps=1.0, min_samples=2, silhouette_score=0.4523443947724053
eps=1.0, min_samples=3, silhouette_score=0.4523443947724053
eps=1.0, min_samples=4, silhouette_score=0.4523443947724053
eps=1.0, min_samples=5, silhouette_score=0.4523443947724053
eps=1.0, min_samples=10, silhouette_score=0.4523443947724053
eps=1.0, min_samples=15, silhouette_score=0.4523443947724053
eps=1.5, min_samples=1, silhouette_score=0.4523443947724053
eps=1.5, min_samples=2, silhouette_score=0.4523443947724053
eps=1.5, min_samples=3, silhouette_score=0.4523443947724053
eps=1.5, min_samples=4, silhouette_score=0.4523443947724053
eps=1.5, min_samples=5, silhouette_score=0.4523443947724053
eps=1.5, min_samples=10, silhouette_score=0.4523443947724053
eps=1.5, min_samples=15, silhouette_score=0.4523443947724053
eps=2.0, min_samples=1, silhouette_score=0.4523443947724053
eps=2.0, min_samples=2, silhouette_score=0.4523443947724053

# DBScan clustering
dbscan = DBSCAN(eps=0.01, min_samples=1)
df['dbscan_label'] = dbscan.fit_predict(X3)

# Visualization of DBScan clustering
trace2 = go.Scatter3d(
    x=df['Age'],
    y=df['Spending Score (1-100)'],
    z=df['Annual Income (k$)'],
    mode='markers',
    marker=dict(
        color=df['dbscan_label'],
        size=20,
        line=dict(
            color=df['dbscan_label'],
            width=12
        ),
        opacity=0.8
    )
)

data_dbscan = [trace2]
layout_dbscan = go.Layout(
    title='DBScan Clusters',
    scene=dict(
        xaxis=dict(title='Age'),
        yaxis=dict(title='Spending Score'),
        zaxis=dict(title='Annual Income')
    )
)

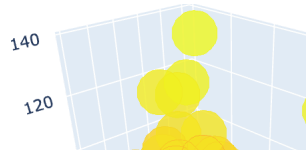
fig_dbscan = go.Figure(data=data_dbscan, layout=layout_dbscan)

py.offline.iplot(fig_dbscan)

```



DBScan Clusters

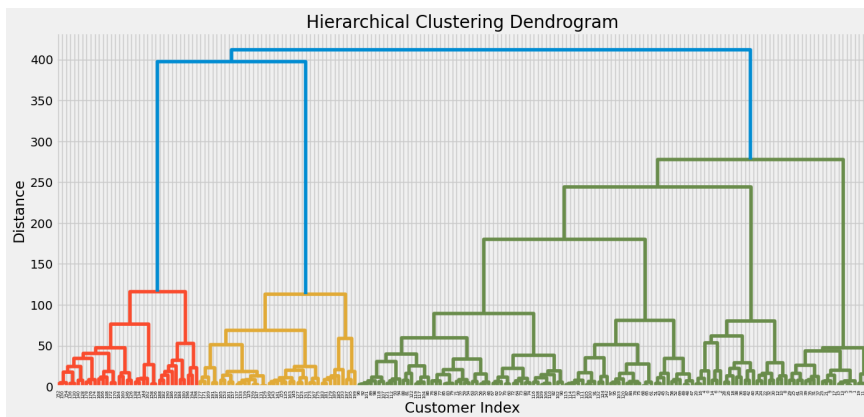


✓ Hierarchical clustering

```
agg_cluster = AgglomerativeClustering(n_clusters=6)
df['agg_label'] = agg_cluster.fit_predict(X3)
```

```
# Calculate linkage matrix
linkage_matrix = sch.linkage(X3, method='ward')
```

```
# Plot the dendrogram
plt.figure(figsize=(15, 7))
dendrogram = sch.dendrogram(linkage_matrix, orientation='top', labels=df.index, distance_sort='descending', show_leaf_counts=True)
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Customer Index')
plt.ylabel('Distance')
plt.show()
```



```
# Visualization of Hierarchical clustering
```

```
trace3 = go.Scatter3d(
    x=df['Age'],
    y=df['Spending Score (1-100)'],
    z=df['Annual Income (k$)'],
    mode='markers',
    marker=dict(
        color=df['agg_label'],
        size=20,
        line=dict(
            color=df['agg_label'],
            width=12
        ),
        opacity=0.8
    )
)
```

```
data_agg = [trace3]
layout_agg = go.Layout(
    title='Hierarchical Clusters',
    scene=dict(
        xaxis=dict(title='Age'),
        yaxis=dict(title='Spending Score'),
```