Finding the best fit clustering technique and analyse Google Play Store Dataset

This project focuses on performing a clustering analysis on the Google Play Store dataset to identify patterns and group similar applications together. The goal is to determine which clustering technique best fits the data set by comparing different algorithms and then analyse playstore data using the particular technique.

```
Start coding or generate with AI.
```

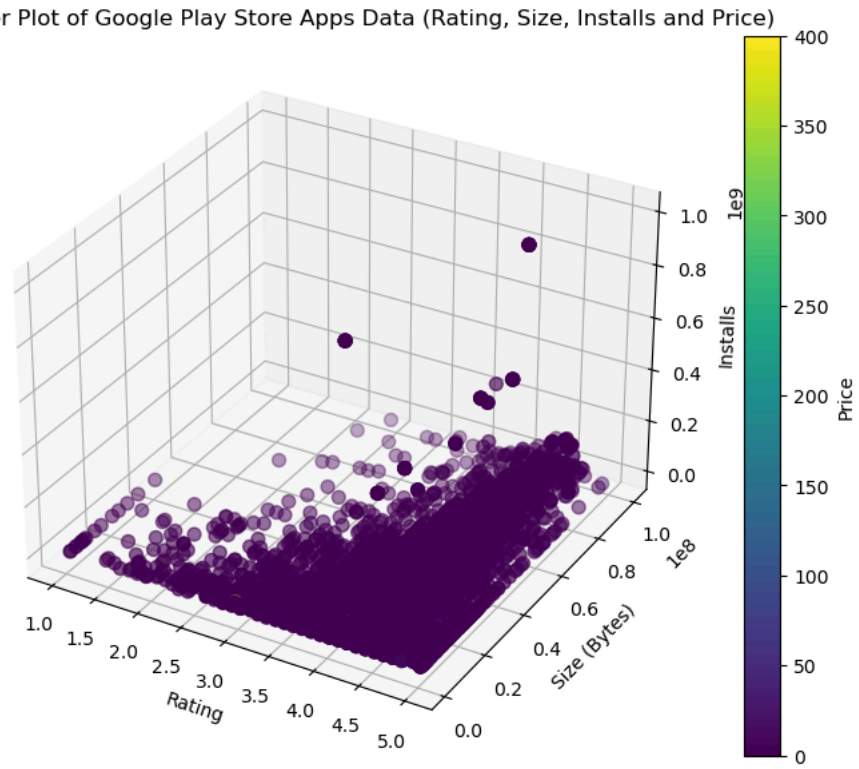
```
# Overview of the data.

import pandas as pd
data= pd.read_csv(r"C:\Users\DELL\Documents\MACHINE LEARNING\googleplaystore.csv")
data.head()
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up
	U Launcher Lite –												

Checking the shape of the data before applying clustering techniques by plotting 3D and pair plot.

```
# Plotting 3-D scatter plot of features (Install, Size, Price and Rating) to get an idea of the data shape.

import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Loading the data from your CSV file
df = pd.read_csv( r'C:\Users\DELL\Documents\MACHINE LEARNING\googleplaystore.csv')

# Removing NaN values from 'Installs' and converting to numeric form
df = df[df['Installs'].str.contains(r'\d', na=False)]
df['Installs'] = df['Installs'].str.replace('[+,]', '', regex=True).astype(int)

# function to convert 'Size' to numeric form
def size_to_numeric(size):
    if 'M' in size:
        return float(size.replace('M', '')) * 1e6
    elif 'K' in size:
        return float(size.replace('K', '')) * 1e3
    return None  # Handle 'Varies with device' or NaN cases

df['Size'] = df['Size'].apply(size_to_numeric)

# Converting 'Price' to numeric form
df['Price'] = df['Price'].str.replace('$', '').replace('Free', '0').astype(float)

# Dropping NaN values rows
df = df.dropna(subset=['Rating', 'Size', 'Installs', 'Price'])

# Plotting a 3D scatter graph with Rating, Size, and Installs
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(df['Rating'], df['Size'], df['Installs'], c=df['Price'], cmap='viridis', s=50)
ax.set_xlabel('Rating')
ax.set_ylabel('Size (Bytes)')
ax.set_zlabel('Installs')
cbar = fig.colorbar(scatter)
cbar.set_label('Price')
plt.title('3D Scatter Plot of Google Play Store Apps Data (Rating, Size, Installs and Price)')
```

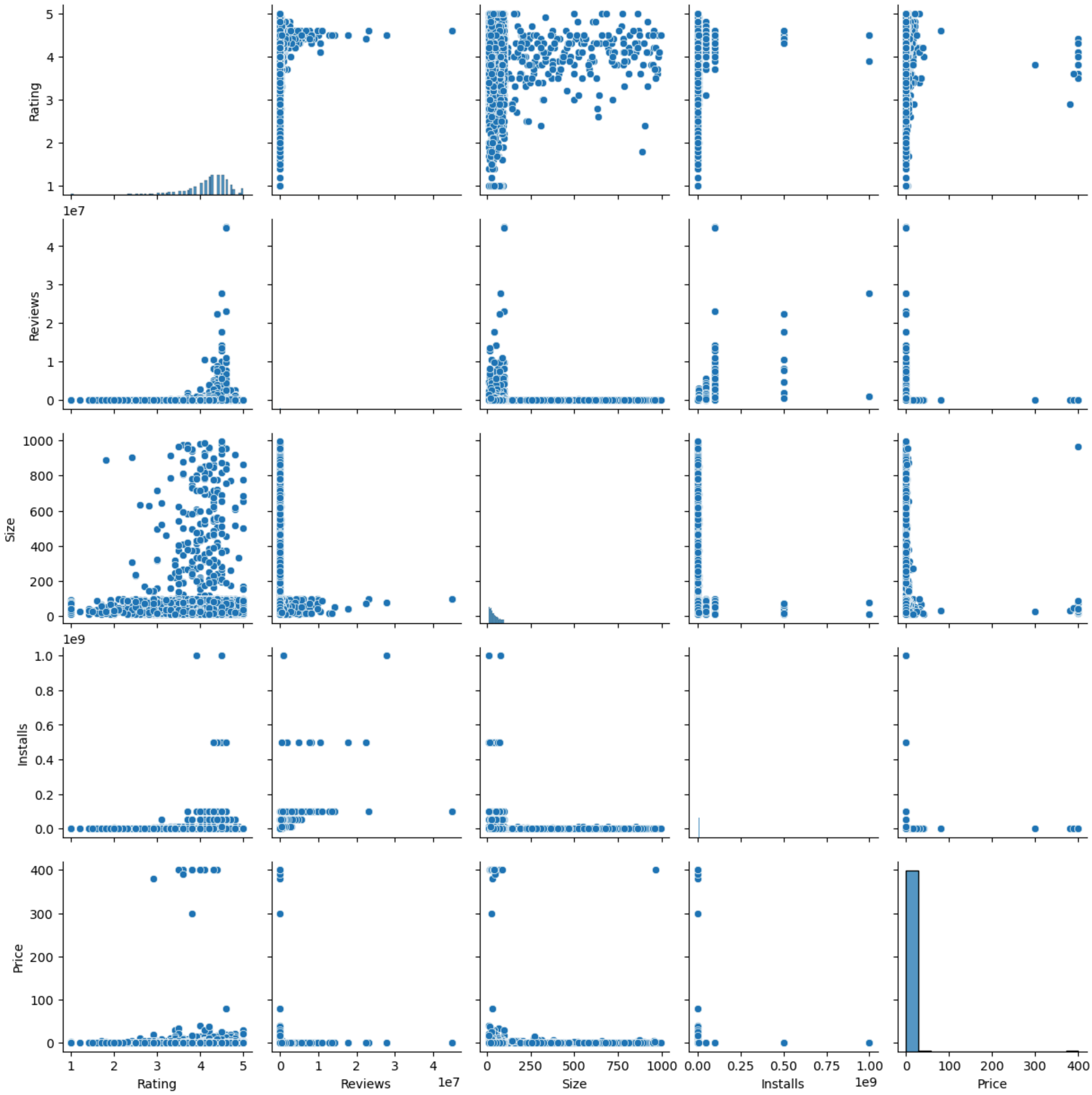
```
plt.show()
```




```
# Plotting a pair plot of various numeric features of Google play store data. (to get the shape of idea in 2-D form also)

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Loading the data
df = pd.read_csv(r"C:\Users\DELL\Documents\MACHINE LEARNING\googleplaystore.csv")

# Converting columns to numeric form
df['Price'] = df['Price'].replace('[\$,]', '', regex=True)
df['Size'] = df['Size'].replace('[\D]', '', regex=True)
df['Reviews'] = pd.to_numeric(df['Reviews'], errors='coerce')
df['Installs'] = pd.to_numeric(df['Installs'].replace('[\+,]', '', regex=True), errors='coerce')

df['Price'] = pd.to_numeric(df['Price'], errors='coerce')
df['Size'] = pd.to_numeric(df['Size'], errors='coerce')

# Dropping NaN value rows
df_numeric = df[['Rating', 'Reviews', 'Size', 'Installs', 'Price']].dropna()

# Ensuring df_numeric is not empty before proceeding
if df_numeric.empty:
    raise ValueError("The DataFrame after cleaning has no data. Check the data cleaning steps.")

# PLotting pair plot
sns.pairplot(df_numeric)
plt.suptitle("Pair Plot of Google Play Store Data - Rating, Review, Size, Install & Price", y=1.02)
plt.show()
```

```
C:\Users\DELL\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be re
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\DELL\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be re
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\DELL\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be re
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\DELL\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be re
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\DELL\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be re
  with pd.option_context('mode.use_inf_as_na', True):
```

Pair Plot of Google Play Store Data - Rating, Review, Size, Install & Price



Start coding or generate with AI.

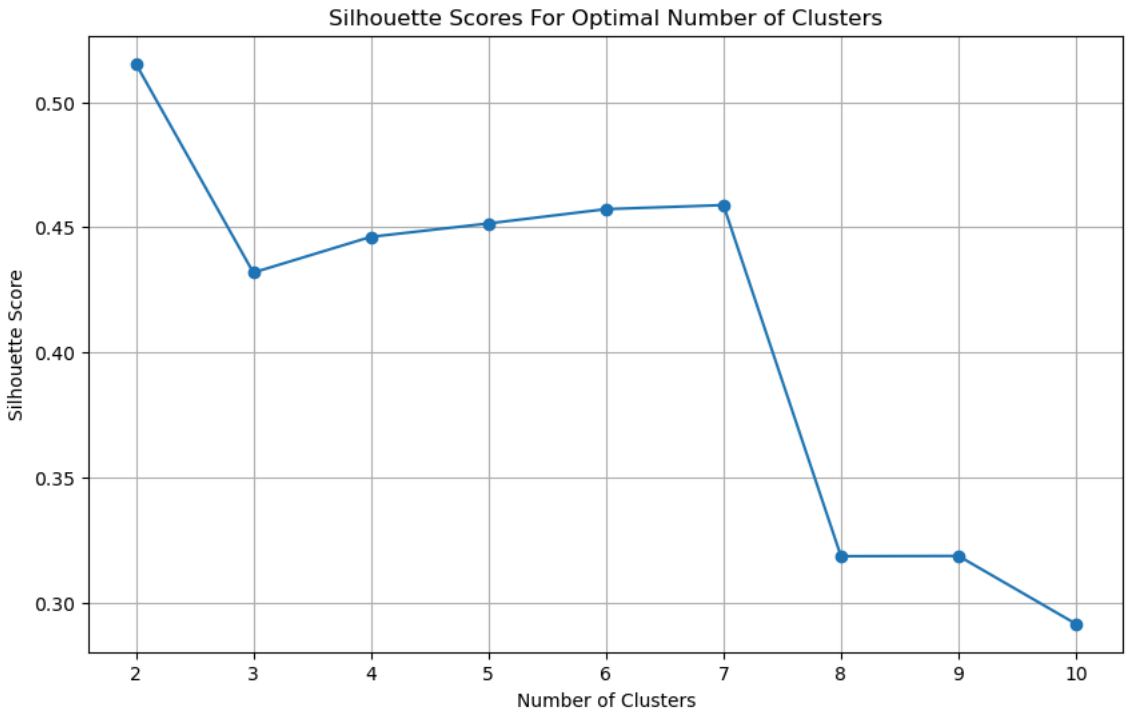
Start coding or generate with AI.

K-MEANS CLUSTERING TECHNIQUE

Start coding or generate with AI.

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt

# Loading the dataset from a CSV file
df = pd.read_csv(r"C:\Users\DELL\Documents\MACHINE LEARNING\googleplaystore.csv")

# Dropping NaN value rows
df.dropna(inplace=True)

# Function to convert 'Size' from strings to numerical values in MB
def convert_size(size):
    if isinstance(size, str):
        if 'M' in size:
            return float(size.replace('M', ''))
        elif 'k' in size:
            return float(size.replace('k', '')) / 1024  # Convert KB to MB
    return np.nan

# Function to convert 'Installs' from strings to numerical values
def convert_installs(installs):
    if isinstance(installs, str):
        try:
            return int(installs.replace(',', '').replace('+', ''))
        except ValueError:
            return np.nan
    return np.nan

# Function to convert 'Price' from strings to numerical values
def convert_price(price):
    if isinstance(price, str):
        try:
            return float(price.replace('$', '')) if price != '0' else 0.0
        except ValueError:
            return np.nan
    return np.nan

# conversions
df['Size'] = df['Size'].apply(convert_size)
df['Installs'] = df['Installs'].apply(convert_installs)
df['Price'] = df['Price'].apply(convert_price)

# Handling NaN values by filling them with column means
df.fillna(df.mean(numeric_only=True), inplace=True)

# features selection
numerical_features = df[['Rating', 'Reviews', 'Size', 'Installs', 'Price']]

# Standardizing the features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(numerical_features)

# Using Silhouette Score method to determine the optimal number of clusters
silhouette_avg = []
kmeans_models = []

for n_clusters in range(2, 11):  # Silhouette score requires at least 2 clusters
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = kmeans.fit_predict(scaled_features)
    silhouette_avg.append(silhouette_score(scaled_features, cluster_labels))
    kmeans_models.append(kmeans)

# Plotting Silhouette Scores
plt.figure(figsize=(10, 6))
plt.plot(range(2, 11), silhouette_avg, marker='o', linestyle='-')
plt.title('Silhouette Scores For Optimal Number of Clusters')
print("")
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
```

```
plt.grid(True)
plt.show()

print('')

# optimal number of clusters based on the highest Silhouette Score
optimal_n_clusters = range(2, 11)[silhouette_avg.index(max(silhouette_avg))]
print(f"Optimal number of clusters: {optimal_n_clusters}")

# building K-means model

# Training the K-Means model
kmeans = KMeans(n_clusters=optimal_n_clusters, random_state=42)
clusters = kmeans.fit_predict(scaled_features)

print('')

# # Getting the cluster centers and centroids
cluster_centers = kmeans.cluster_centers_

print('')
# original scale of the cluster centers
original_centers = scaler.inverse_transform(cluster_centers)

# Adding cluster to the original DataFrame
df['Cluster'] = clusters

print('')
# printing DataFrame withcluster
print(df.head())

print('')

# final cluster centers and centroids
print(f"Final cluster centers (in original scale): \n{original_centers}")
print('')
print(f"Final cluster centroids (in standardized scale): \n{cluster_centers}")
```

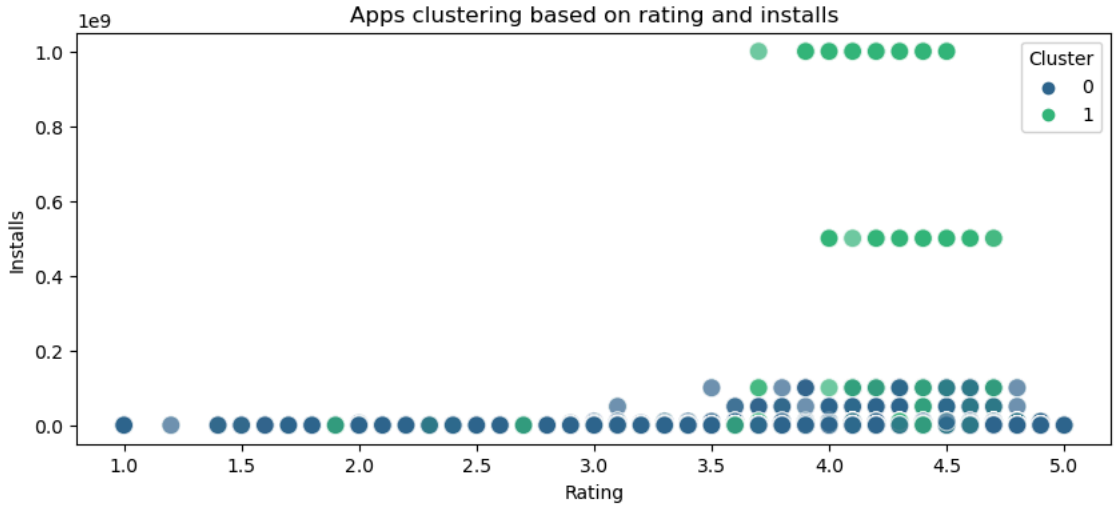



```
Optimal number of clusters: 2
```

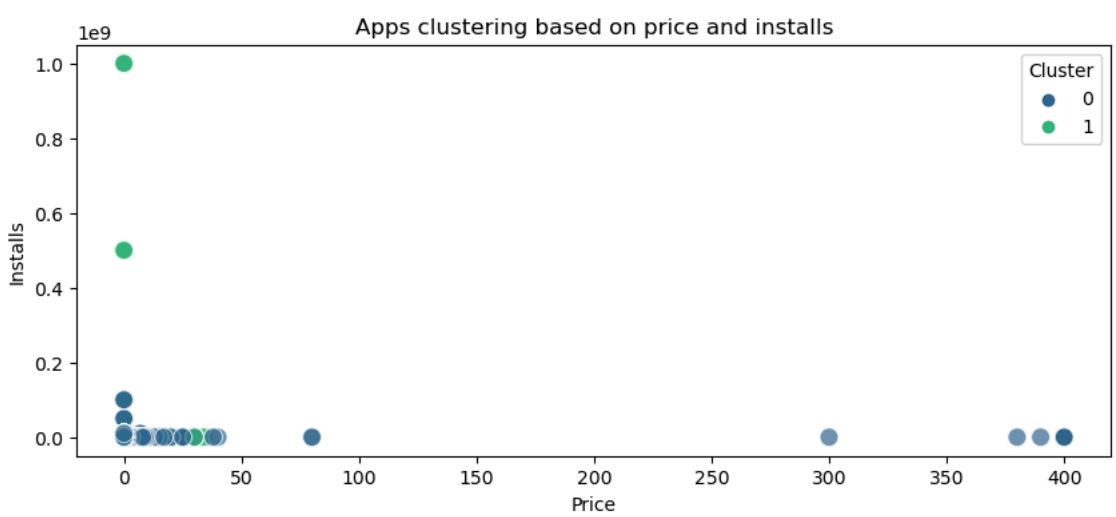
	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver	Android Ver	Cluster
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19.0	10000	Free	0.0	Everyone	Art & Design	January 7, 2018	1.0.0	4.0.3 and up	0
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14.0	500000	Free	0.0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0	4.0.3 and up	0
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7	5000000	Free	0.0	Everyone	Art & Design	August 1, 2018	1.2.4	4.0.3 and up	0
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25.0	50000000	Free	0.0	Teen	Art & Design	June 8, 2018	Varies with device	4.2 and up	0
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8	100000	Free	0.0	Everyone	Art & Design;Creativity	June 20, 2018	1.1	4.4 and up	0

```
Final cluster centers (in original scale):
[[4.17129259e+00 1.57958329e+05 1.60919558e+01 5.98224767e+06
  1.08349198e+00]
 [4.31104651e+00 2.58243217e+06 6.28794662e+01 8.71101937e+07
  2.52158430e-01]]

Final cluster centroids (in standardized scale):
[[-0.03987502 -0.11333379 -0.32292646 -0.13068493  0.00772484]
 [ 0.23136783  0.65759957  1.87372447  0.75827652 -0.04482205]]
```

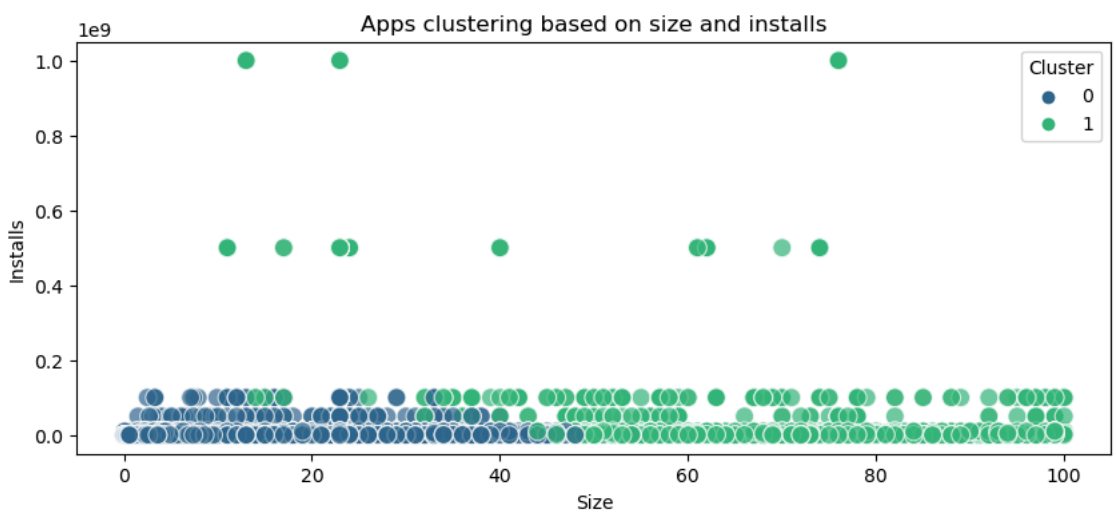
```
# plotting cluster for rating and installs features

import seaborn as sns
plt.figure(figsize=(10, 4))
sns.scatterplot(x=df['Rating'], y=df['Installs'], hue=df['Cluster'], palette='viridis', s=100, alpha=0.7)
plt.title('Apps clustering based on rating and installs')
plt.show()
```




```
# plotting cluster for price and installs features

import seaborn as sns
plt.figure(figsize=(10, 4))
sns.scatterplot(x=df['Price'], y=df['Installs'], hue=df['Cluster'], palette='viridis', s=100, alpha=0.7)
plt.title('Apps clustering based on price and installs')
plt.show()
```




```
# plotting cluster for size and installs features

import seaborn as sns
plt.figure(figsize=(10, 4))
sns.scatterplot(x=df['Size'], y=df['Installs'], hue=df['Cluster'], palette='viridis', s=100, alpha=0.7)
plt.title('Apps clustering based on size and installs')
plt.show()
```



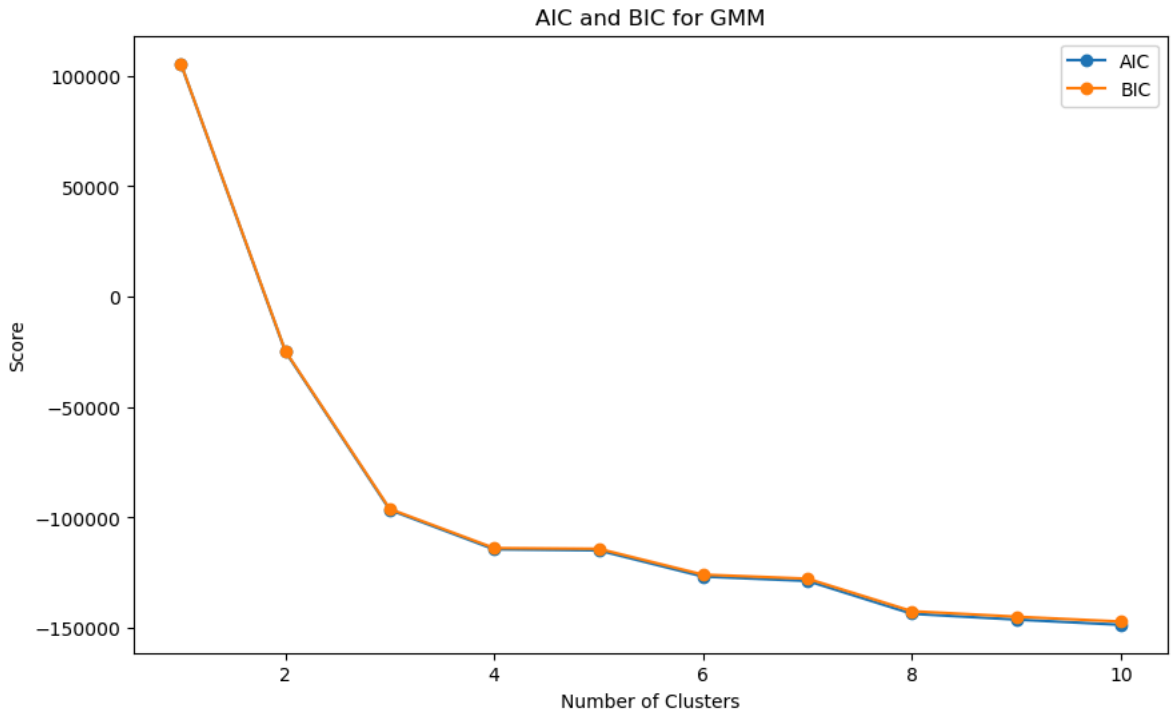
Start coding or generate with AI.

Start coding or generate with AI.

GMM CLUSTERING TECHNIQUE

Start coding or generate with AI.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.mixture import GaussianMixture
import numpy as np
import matplotlib.pyplot as plt

# Loading the data from CSV
df = pd.read_csv(r"C:\Users\DELL\Documents\MACHINE LEARNING\googleplaystore.csv")

# function to convert 'Size' to numeric form
def convert_size(size_str):
    if 'M' in size_str:
        return float(size_str.replace('M', '')) * 1e6
    elif 'k' in size_str:
        return float(size_str.replace('k', '')) * 1e3
    return np.nan

df['Size'] = df['Size'].apply(convert_size)

# Converting 'Installs' from string to numeric form
df['Installs'] = df['Installs'].str.replace('+', '').str.replace(',', '')

# Handle non-numeric values in installs
df['Installs'] = pd.to_numeric(df['Installs'], errors='coerce')
df = df.dropna(subset=['Installs'])

# function to convert 'Price' to numeric form
def convert_price(price_str):
    return float(price_str.replace('$', '')) if price_str != '0' else 0.0

df['Price'] = df['Price'].apply(convert_price)

# Selecting the features for clustering and dropping NaN rows
X = df[['Rating', 'Reviews', 'Size', 'Installs', 'Price']].dropna()

# Standardizing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Function to calculate AIC and BIC to find optimal number of clusters.
def calculate_aic_bic(X, max_clusters=10):
    aic = []
    bic = []
    for n_clusters in range(1, max_clusters + 1):
        gmm = GaussianMixture(n_components=n_clusters, random_state=42)
        gmm.fit(X)
        aic.append(gmm.aic(X))
        bic.append(gmm.bic(X))

    return aic, bic

aic, bic = calculate_aic_bic(X_scaled, max_clusters=10)

# Plotting AIC and BIC scores
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), aic, label='AIC', marker='o')
plt.plot(range(1, 11), bic, label='BIC', marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
plt.title('AIC and BIC for GMM')
plt.legend()
plt.show()

# optimal number of clusters based on the lowest AIC and BIC
optimal_n_clusters_aic = np.argmin(aic) + 1
optimal_n_clusters_bic = np.argmin(bic) + 1

print(f"Optimal number of clusters based on AIC: {optimal_n_clusters_aic}")
print(f"Optimal number of clusters based on BIC: {optimal_n_clusters_bic}")
```




```
Optimal number of clusters based on AIC: 10
Optimal number of clusters based on BIC: 10
```

```
# optimal number of clusters based on lowest score
optimal_n_clusters = 10

# Creating GMM Model
gmm = GaussianMixture(n_components=optimal_n_clusters, random_state=42)
df['Cluster'] = np.nan  # Initialize the Cluster column
cluster_labels = gmm.fit_predict(X_scaled)

# adding cluster to dataframe
df.loc[X.index, 'Cluster'] = cluster_labels

# Remove rows with NaN in the 'Cluster' column
df = df.dropna(subset=['Cluster'])

# printing the dataframe with cluster columns
print(df.head())
```

```
                                                 App        Category  Rating  \
0     Photo Editor & Candy Camera & Grid & ScrapBook  ART_AND_DESIGN     4.1
1                               Coloring book moana  ART_AND_DESIGN     3.9
2  U Launcher Lite – FREE Live Cool Themes, Hide ...  ART_AND_DESIGN     4.7
3                             Sketch - Draw & Paint  ART_AND_DESIGN     4.5
4              Pixel Draw - Number Art Coloring Book  ART_AND_DESIGN     4.3

  Reviews        Size    Installs  Type  Price Content Rating  \
0     159  19000000.0     10000.0  Free    0.0       Everyone
1     967  14000000.0    500000.0  Free    0.0       Everyone
2   87510   8700000.0   5000000.0  Free    0.0       Everyone
3  215644  25000000.0  50000000.0  Free    0.0           Teen
4     967   2800000.0    100000.0  Free    0.0       Everyone

                     Genres      Last Updated         Current Ver  \
0              Art & Design   January 7, 2018               1.0.0
1  Art & Design;Pretend Play  January 15, 2018               2.0.0
2              Art & Design    August 1, 2018               1.2.4
3              Art & Design      June 8, 2018  Varies with device
4    Art & Design;Creativity     June 20, 2018                 1.1

    Android Ver  Cluster
0  4.0.3 and up      9.0
1  4.0.3 and up      0.0
2  4.0.3 and up      1.0
3    4.2 and up      6.0
4    4.4 and up      9.0
```

```
# checking number of unique clusters.

np.unique(df['Cluster'])
```

```
array([0., 1., 2., 3., 4., 5., 6., 7., 8., 9.])
```

```
np.bincount(df['Cluster'])
```

```
array([1755, 1282,   90,   58,   71,  418,  299,  195,    4, 3557],
      dtype=int64)
```

```
# Displaying means of each clusters

cluster_means = gmm.means_
print("\nCluster Means:")
print(cluster_means)
```

```
Cluster Means:
[[ 3.82489603e-02 -1.47746887e-01  1.90014418e-01 -1.51659232e-01
  -6.48047756e-02]
 [ 1.75236393e-01 -7.63957851e-02  3.23213495e-01 -2.26157724e-02
  -6.48047756e-02]
 [-3.13530330e-01 -1.57680848e-01 -2.07296203e-01 -1.67604295e-01
   4.33880210e+00]
 [ 3.81905943e-01  5.15622009e+00  1.02550058e+00  8.58829442e+00
  -6.48047756e-02]
 [ 5.93191465e-01 -1.31472882e-01  4.91579318e-01 -1.56718582e-01
   1.42812086e-01]
 [ 1.70617193e-01 -1.57759412e-01 -1.73968889e-01 -1.67663095e-01
   1.02718019e-01]
 [ 4.28455324e-01  3.38705520e-01  7.94415501e-01  3.71147462e-01
  -6.48047756e-02]
 [ 4.31772949e-01  2.56856171e+00  1.21447415e+00  1.82670542e+00
  -6.48047756e-02]
 [ 7.82602253e-01  2.39358413e+01  3.20092456e+00  1.82670542e+00
  -6.48047756e-02]
 [-1.74820547e-01 -1.57794296e-01 -3.52752414e-01 -1.67220000e-01
  -6.48047756e-02]]
```

```
# Displaying covariance of each cluster.

cluster_covariances = gmm.covariances_
print("\nCluster Covariances:")
print(cluster_covariances)
```

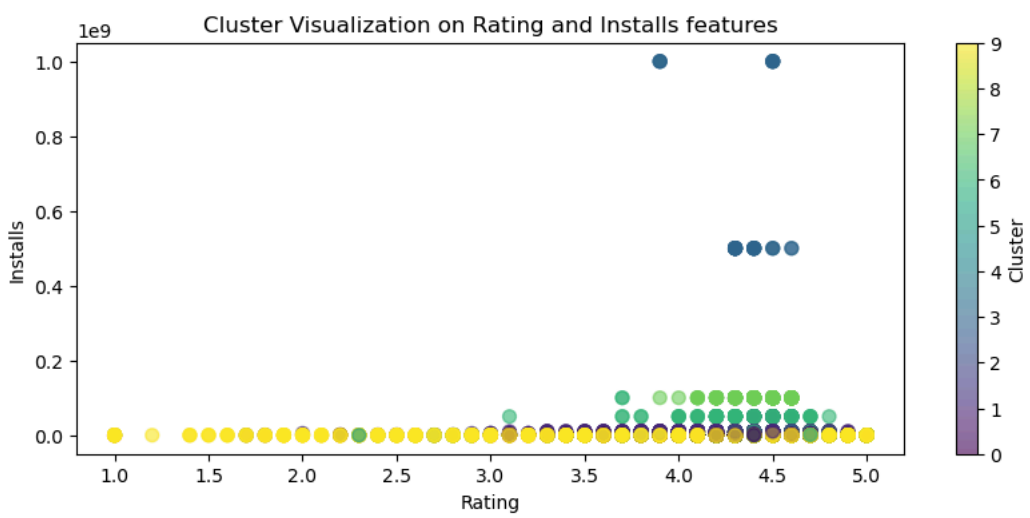
```
Cluster Covariances:
[[[ 5.05442944e-01  1.94667152e-03  2.75452106e-02 -9.75688492e-06
    -1.30942572e-32]
  [ 1.94667152e-03  1.00178507e-04  9.66711546e-05  2.36179067e-05
    2.81547559e-32]
  [ 2.75452106e-02  9.66711546e-05  1.19831408e+00 -1.36355989e-03
   -2.96092839e-32]
  [-9.75688492e-06  2.36179067e-05 -1.36355989e-03  3.28205225e-05
    3.50711572e-32]
  [-1.00660350e-32  2.81438031e-32 -2.80951728e-32  3.50743883e-32
    1.00000000e-06]]

 [[ 3.29601696e-01  1.08841367e-02  5.64155835e-02 -1.51810926e-03
   -9.00043168e-32]
  [ 1.08841367e-02  3.94578492e-03  1.26622488e-02  1.44160628e-03
    4.01261983e-32]
  [ 5.64155835e-02  1.26622488e-02  1.19551332e+00 -6.75979275e-04
   -1.71873963e-31]
  [-1.51810926e-03  1.44160628e-03 -6.75979275e-04  3.53202992e-03
    1.18025351e-32]
  [-8.76801253e-32  4.00293569e-32 -1.66644532e-31  1.19695864e-32
    1.00000000e-06]]

 [[ 2.04097512e+00  2.29243643e-04  2.48451246e-01  1.22241177e-04
   -5.47555971e-01]
  [ 2.29243643e-04  2.09079164e-06 -1.63856641e-05  4.25374116e-07
   -5.63502317e-04]
  [ 2.48451246e-01 -1.63856641e-05  7.18715895e-01 -1.00866238e-05
   -1.16777760e+00]
  [ 1.22241177e-04  4.25374116e-07 -1.00866238e-05  1.33077498e-06
   -1.56204720e-05]
  [-5.47555971e-01 -5.63502317e-04 -1.16777760e+00 -1.56204720e-05
    6.24957887e+01]]

 [[ 8.97680628e-02  4.59505123e-01  9.13196236e-02 -8.02420614e-01
   -2.36414318e-30]
  [ 4.59505123e-01  2.28524277e+01  2.81899192e+00  1.60990304e+01
   -3.05018514e-29]
  [ 9.13196236e-02  2.81899192e+00  1.22768004e+00  7.34472025e-01
   -6.50264632e-30]
  [-8.02420614e-01  1.60990304e+01  7.34472025e-01  4.20703583e+01
   -5.23962702e-29]
  [-2.37211258e-30 -3.06226513e-29 -6.45566710e-30 -5.23426119e-29
    1.00000000e-06]]

 [[ 1.28753680e-01 -4.27865728e-04 -6.77844828e-02 -4.38840155e-04
```

```
  2.99830528e-03]
 [-4.27865728e-04  1.27925997e-03  3.61345431e-03  6.20933862e-04
  -5.95498225e-04]
 [-6.77844828e-02  3.61345431e-03  1.42299490e+00  1.58241905e-03
  -3.30862799e-02]
 [-4.38840155e-04  6.20933862e-04  1.58241905e-03  5.45476303e-04
  -6.97951025e-04]
 [ 2.99830528e-03 -5.95498225e-04 -3.30862799e-02 -6.97951025e-04
   1.56870288e-02]]

[[ 8.86672192e-01  8.18012799e-05  6.54005010e-02  2.29375222e-05
```

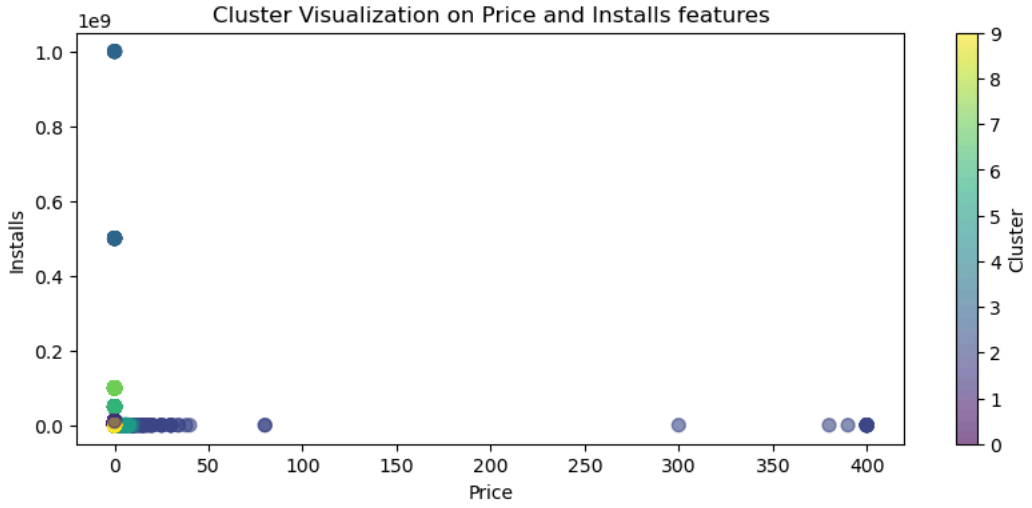
```
# Displaying weights of each cluster.

Cluster_Weights = gmm.weights_
print("\nCluster Weights:")
print(Cluster_Weights)
```

```
Cluster Weights:
[0.22750616 0.16467834 0.01223139 0.00760296 0.00965237 0.05335708
 0.0406375  0.02523464 0.00051753 0.45858203]
```

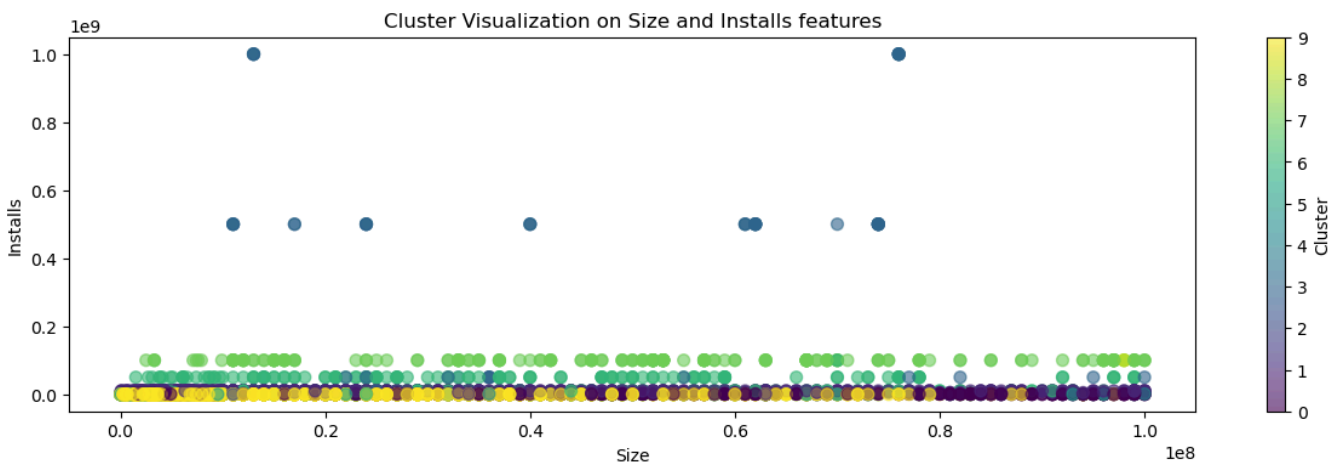
```
# Visualizing the clusters using two selected features -'Rating' and 'Installs'

plt.figure(figsize=(10, 4))
plt.scatter(df['Rating'], df['Installs'], c=cluster_labels, cmap='viridis', s=50, alpha=0.6)
plt.colorbar(label='Cluster')
plt.xlabel('Rating')
plt.ylabel('Installs')
plt.title('Cluster Visualization on Rating and Installs features')
plt.show()
```




```
# Visualizing the clusters using two selected features -'Price' and 'Installs'

plt.figure(figsize=(10, 4))
plt.scatter(df['Price'], df['Installs'], c=cluster_labels, cmap='viridis', s=50, alpha=0.6)
plt.colorbar(label='Cluster')
plt.xlabel('Price')
plt.ylabel('Installs')
plt.title('Cluster Visualization on Price and Installs features')
plt.show()
```




```
# Visualizing the clusters using two selected features -'Size' and 'Installs'

plt.figure(figsize=(15, 4))
plt.scatter(df['Size'], df['Installs'], c=cluster_labels, cmap='viridis', s=50, alpha=0.6)
plt.colorbar(label='Cluster')
plt.xlabel('Size')
plt.ylabel('Installs')
plt.title('Cluster Visualization on Size and Installs features')
plt.show()
```



Start coding or generate with AI.

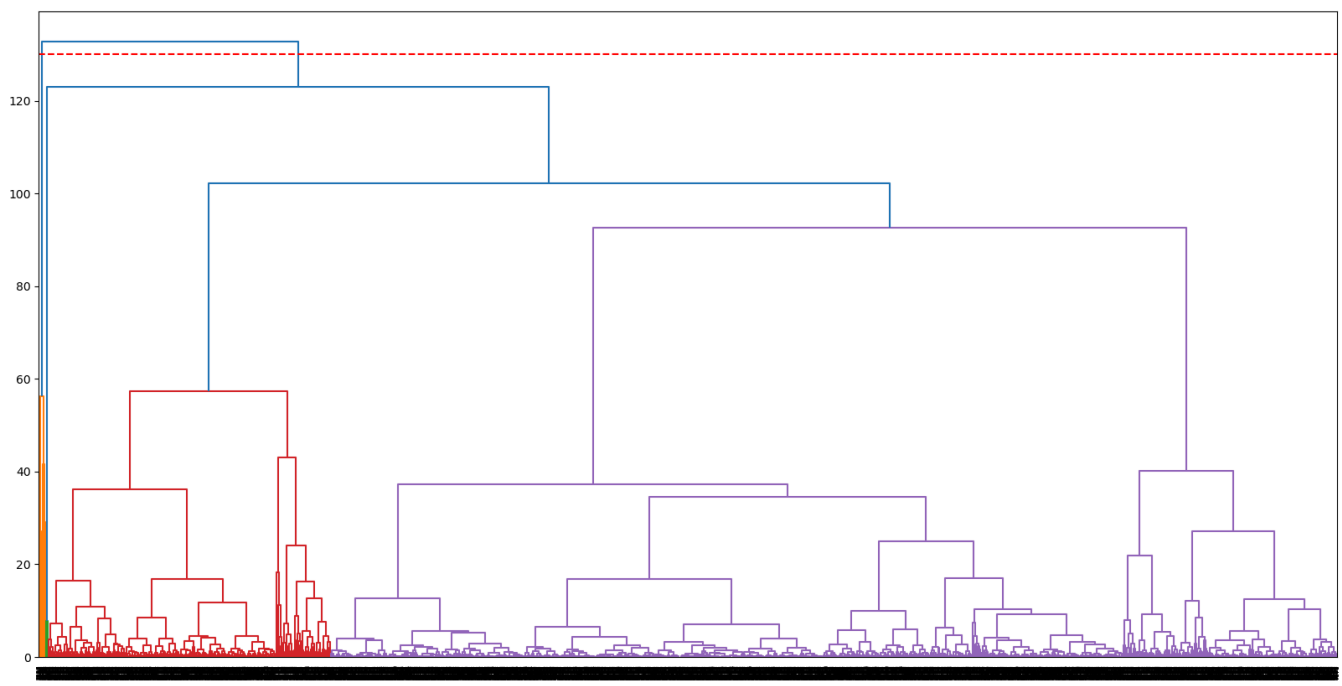
Start coding or generate with AI.

HIERARCHICAL CLUSTERING TECHNIQUE

Start coding or generate with AI.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
import numpy as np
import matplotlib.pyplot as plt

# Loading the data from CSV
df = pd.read_csv(r"C:\Users\DELL\Documents\MACHINE LEARNING\googleplaystore.csv")

# Function to convert 'Size' to numeric form
def convert_size(size_str):
    if 'M' in size_str:
        return float(size_str.replace('M', '')) * 1e6
    elif 'k' in size_str:
        return float(size_str.replace('k', '')) * 1e3
```

```
    return np.nan

df['Size'] = df['Size'].apply(convert_size)

# Converting 'Installs' from string to numeric form
df['Installs'] = df['Installs'].str.replace('+', '').str.replace(',', '')

df['Installs'] = pd.to_numeric(df['Installs'], errors='coerce')
df = df.dropna(subset=['Installs'])

# Function to convert 'Price' to numeric form
def convert_price(price_str):
    return float(price_str.replace('$', '')) if price_str != '0' else 0.0

df['Price'] = df['Price'].apply(convert_price)

# Selecting features for clustering and dropping NaN value rows
X = df[['Rating', 'Reviews', 'Size', 'Installs', 'Price']].dropna()

# Standardizing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)



# plotting dendogram
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(20,10));

dend=shc.dendrogram(shc.linkage(X_scaled,method='ward'));

plt.axhline(y=130, color='red',linestyle='--');
```




```
# developing hierarchical - agglomerative clustering model.

from sklearn.cluster import AgglomerativeClustering

X_scaled = pd.DataFrame(X_scaled).dropna()
X_scaled = X_scaled.values

clust = AgglomerativeClustering(n_clusters=2, linkage='ward')
clusters = clust.fit_predict(X_scaled)


clusters
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

```
# checking the unique clusters

unique_clusters = np.unique(clusters)
print(f"Unique clusters: {unique_clusters}")
```

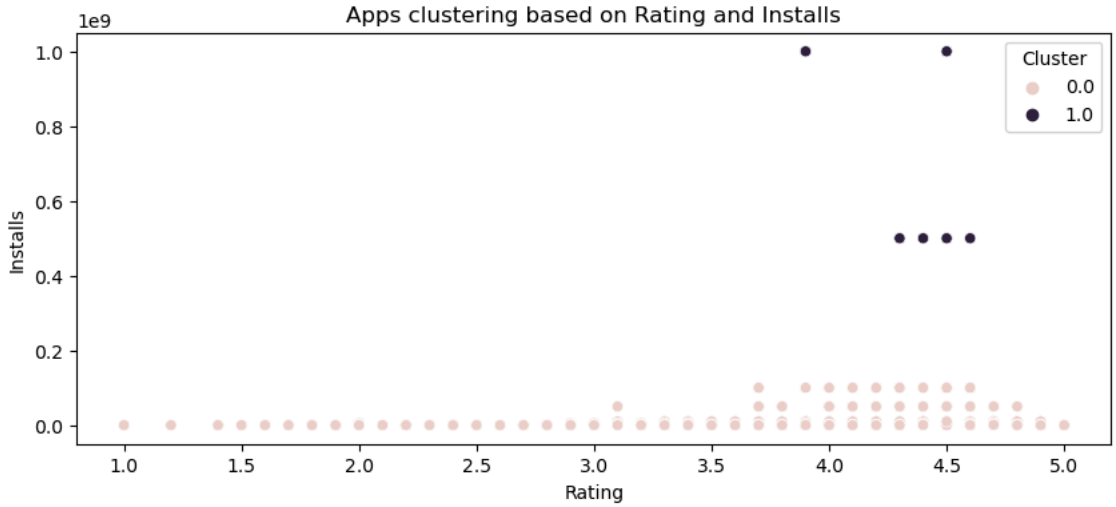
```
Unique clusters: [0 1]
```

```
# Adding clusters to the original DataFrame
df.loc[X.index, 'Cluster'] = clusters

# printing the resulting DataFrame with cluster labels
print(df.head())
```

```
                                                 App        Category  Rating  \
0     Photo Editor & Candy Camera & Grid & ScrapBook  ART_AND_DESIGN     4.1
1                                 Coloring book moana  ART_AND_DESIGN     3.9
2  U Launcher Lite – FREE Live Cool Themes, Hide ...  ART_AND_DESIGN     4.7
3                               Sketch - Draw & Paint  ART_AND_DESIGN     4.5
4              Pixel Draw - Number Art Coloring Book  ART_AND_DESIGN     4.3

  Reviews        Size    Installs  Type  Price Content Rating  \
0     159  19000000.0     10000.0  Free    0.0       Everyone
1     967  14000000.0    500000.0  Free    0.0       Everyone
2   87510   8700000.0   5000000.0  Free    0.0       Everyone
3  215644  25000000.0  50000000.0  Free    0.0           Teen
4     967   2800000.0    100000.0  Free    0.0       Everyone

                      Genres      Last Updated         Current Ver  \
0               Art & Design   January 7, 2018               1.0.0
1  Art & Design;Pretend Play  January 15, 2018               2.0.0
2               Art & Design    August 1, 2018               1.2.4
3               Art & Design      June 8, 2018  Varies with device
4    Art & Design;Creativity     June 20, 2018                 1.1

    Android Ver  Cluster
0  4.0.3 and up      0.0
1  4.0.3 and up      0.0
2  4.0.3 and up      0.0
3    4.2 and up      0.0
4    4.4 and up      0.0
```

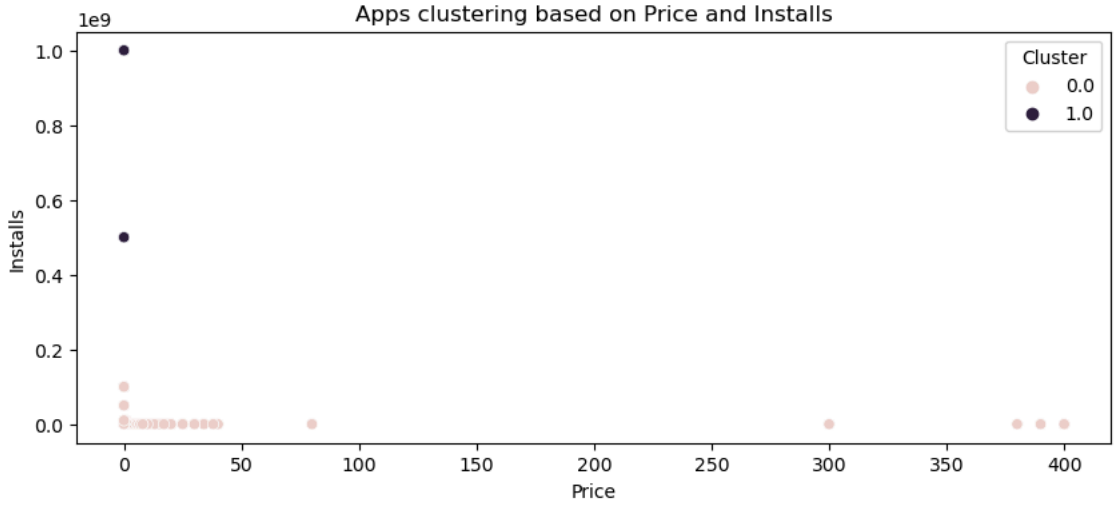
```
# plotting the App cluster for Installs and Rating

import seaborn as sns

plt.figure(figsize=(10,4));
sns.scatterplot(x=df['Rating'],y=df['Installs'],hue=df['Cluster']);
plt.title('Apps clustering based on Rating and Installs')
```

```
Text(0.5, 1.0, 'Apps clustering based on Rating and Installs')
```




```
# plotting the App cluster for Installs and Price

import seaborn as sns

plt.figure(figsize=(10,4));
sns.scatterplot(x=df['Price'],y=df['Installs'],hue=df['Cluster']);
plt.title('Apps clustering based on Price and Installs')
```

```
Text(0.5, 1.0, 'Apps clustering based on Price and Installs')
```



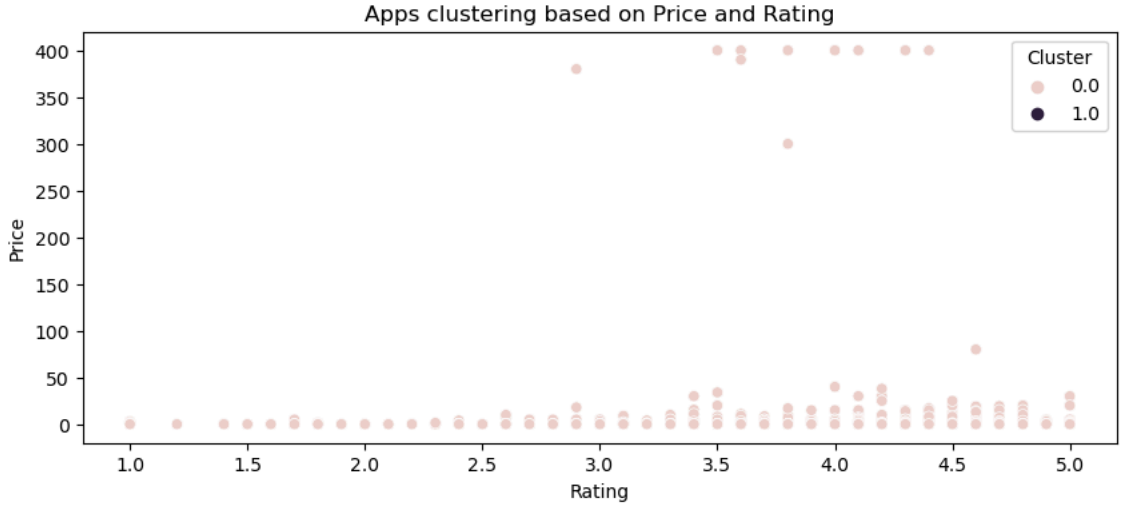

```
# plotting the App cluster for Installs and Size

import seaborn as sns
plt.figure(figsize=(10,4));
sns.scatterplot(x=df['Size'],y=df['Installs'],hue=df['Cluster']);
plt.title('Apps clustering based on Size and Installs')
```

```
Text(0.5, 1.0, 'Apps clustering based on Size and Installs')
```


```
Start coding or generate with AI.
```

```
# plotting the App cluster for Price and Rating

import seaborn as sns
plt.figure(figsize=(10,4));
sns.scatterplot(x=df['Rating'],y=df['Price'],hue=df['Cluster']);
plt.title('Apps clustering based on Price and Rating')
```

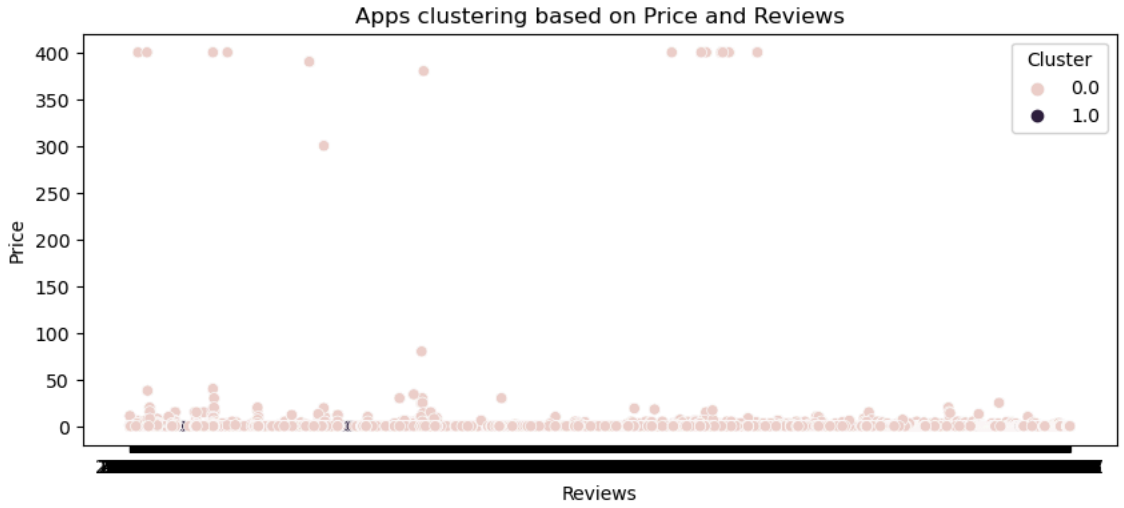
```
Text(0.5, 1.0, 'Apps clustering based on Price and Rating')
```




```
Start coding or generate with AI.
```

```
# plotting the App cluster for Price and Reviews

import seaborn as sns
plt.figure(figsize=(10,4));
sns.scatterplot(x=df['Reviews'],y=df['Price'],hue=df['Cluster']);
plt.title('Apps clustering based on Price and Reviews')
```

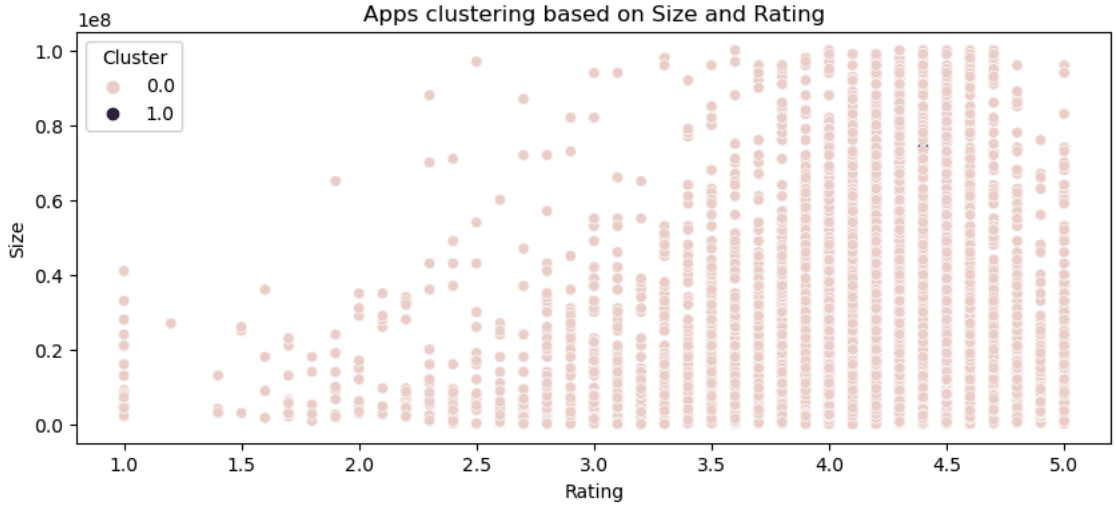
```
Text(0.5, 1.0, 'Apps clustering based on Price and Reviews')
```




```
Start coding or generate with AI.
```

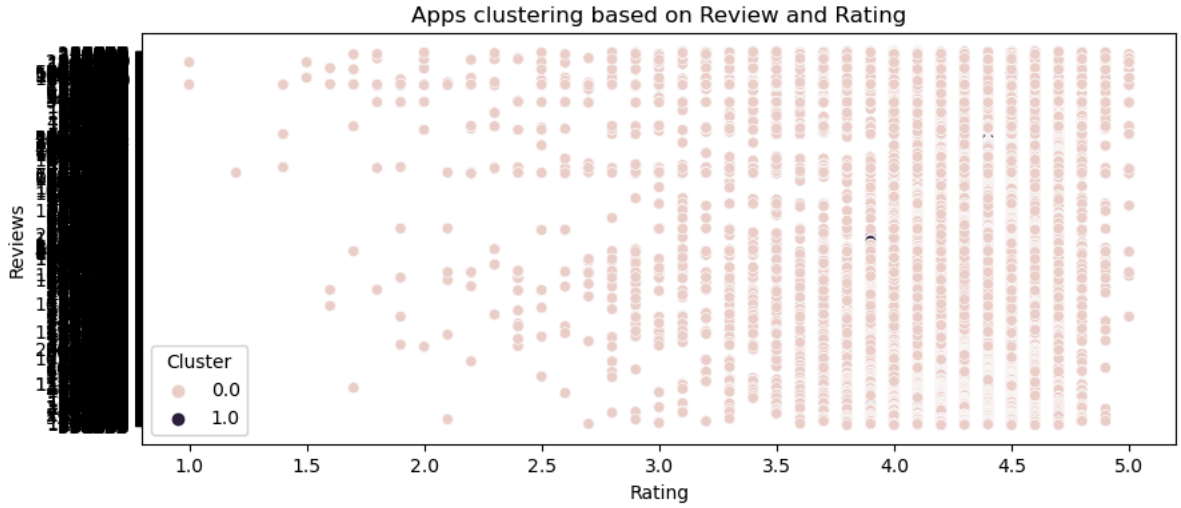
```
# plotting the App cluster for Size and rating

import seaborn as sns
plt.figure(figsize=(10,4));
sns.scatterplot(x=df['Rating'],y=df['Size'],hue=df['Cluster']);
plt.title('Apps clustering based on Size and Rating')
```

```
Text(0.5, 1.0, 'Apps clustering based on Size and Rating')
```




```
# plotting the App cluster for Reviews and rating

import seaborn as sns
plt.figure(figsize=(10,4));
sns.scatterplot(x=df['Rating'],y=df['Reviews'],hue=df['Cluster']);
plt.title('Apps clustering based on Review and Rating')
```

```
Text(0.5, 1.0, 'Apps clustering based on Review and Rating')
```




```
Start coding or generate with AI.
```

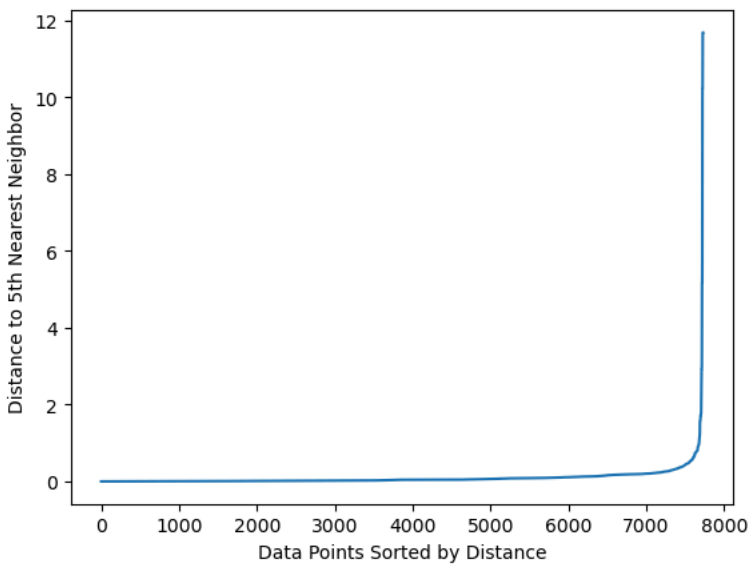
```
Start coding or generate with AI.
```

DBSCAN CLUSTERING TECHNIQUE

```
Start coding or generate with AI.
```

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN
import numpy as np
import matplotlib.pyplot as plt

# Loading the data from CSV
df = pd.read_csv(r"C:\Users\DELL\Documents\MACHINE LEARNING\googleplaystore.csv")

# Function to convert 'Size' to numeric form
def convert_size(size_str):
    if 'M' in size_str:
        return float(size_str.replace('M', '')) * 1e6
    elif 'k' in size_str:
        return float(size_str.replace('k', '')) * 1e3
    return np.nan
```

```
df['Size'] = df['Size'].apply(convert_size)

# Converting 'Installs' from string to numeric form
df['Installs'] = df['Installs'].str.replace('+', '').str.replace(',', '')
df['Installs'] = pd.to_numeric(df['Installs'], errors='coerce')
df = df.dropna(subset=['Installs'])

# function convert 'Price' to numeric form
def convert_price(price_str):
    return float(price_str.replace('$', '')) if price_str != '0' else 0.0

df['Price'] = df['Price'].apply(convert_price)

# Selecting features for clustering and dropping NaN value rows.
X = df[['Rating', 'Reviews', 'Size', 'Installs', 'Price']].dropna()

# Standardizing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)



# k nearest neighbour method to find optimal epsilon value.

from sklearn.neighbors import NearestNeighbors
import matplotlib.pyplot as plt
import numpy as np

neighbors = NearestNeighbors(n_neighbors=5)   # as we have taken 5 features for clustering
neighbors_fit = neighbors.fit(X_scaled)
distances, indices = neighbors_fit.kneighbors(X_scaled)

# Sorting distances.
distances = np.sort(distances[:, 4], axis=0)  # use 4 because we used 5 neighbors

# plotting distance and data points.
plt.plot(distances)
plt.ylabel('Distance to 5th Nearest Neighbor')
plt.xlabel('Data Points Sorted by Distance')
plt.show()
```



Unsupported Cell Type. Double-Click to inspect/edit the content.

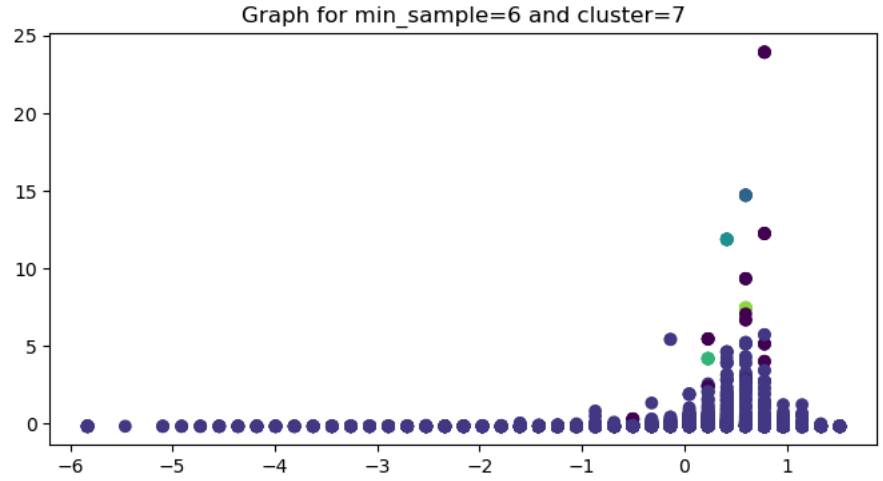
Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# min_samples = 6

dbscan=DBSCAN(eps=1,min_samples=6)
dbscan.fit(X_scaled)
set(dbscan.labels_)
```

```
{-1, 0, 1, 2, 3, 4, 5}
```

```
plt.figure(figsize=(8,4))
plt.scatter(X_scaled[:,0],X_scaled[:,1],c=dbscan.labels_);
plt.title('Graph for min_sample=6 and cluster=7')
```

```
Text(0.5, 1.0, 'Graph for min_sample=6 and cluster=7')
```

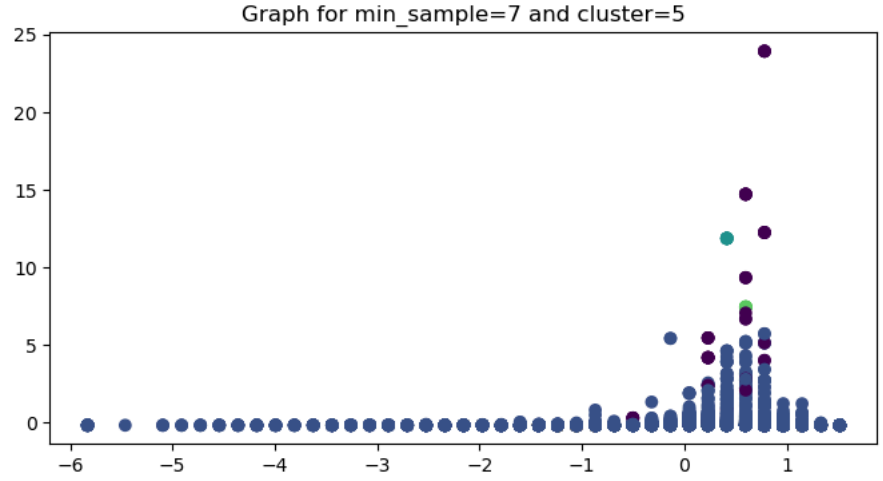



```
# min_samples = 7

dbscan=DBSCAN(eps=1,min_samples=7) # eps
dbscan.fit(X_scaled)
set(dbscan.labels_)
```

```
{-1, 0, 1, 2, 3}
```

```
plt.figure(figsize=(8,4))
plt.scatter(X_scaled[:,0],X_scaled[:,1],c=dbscan.labels_);
plt.title('Graph for min_sample=7 and cluster=5')
```

```
Text(0.5, 1.0, 'Graph for min_sample=7 and cluster=5')
```

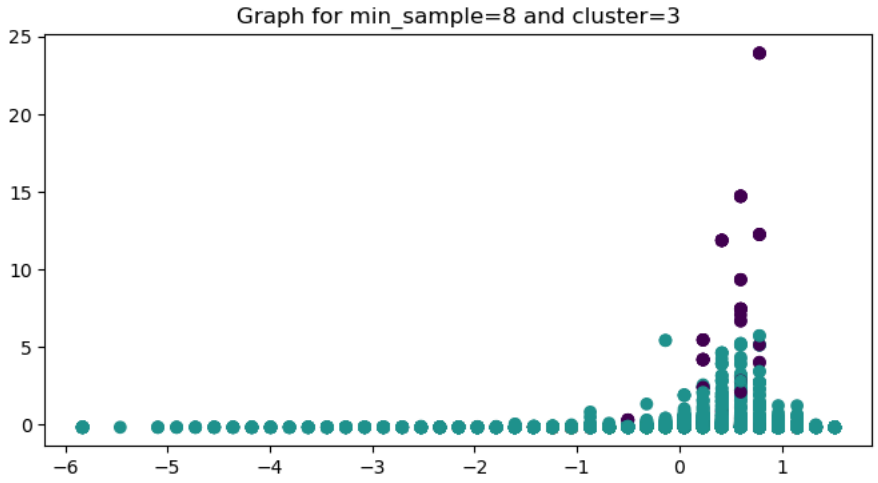



```
# min_samples = 8

dbscan=DBSCAN(eps=1,min_samples=8) # eps
dbscan.fit(X_scaled)
set(dbscan.labels_)
```

```
{-1, 0, 1}
```

```
plt.figure(figsize=(8,4))
plt.scatter(X_scaled[:,0],X_scaled[:,1],c=dbscan.labels_);
plt.title('Graph for min_sample=8 and cluster=3')
```

```
Text(0.5, 1.0, 'Graph for min_sample=8 and cluster=3')
```

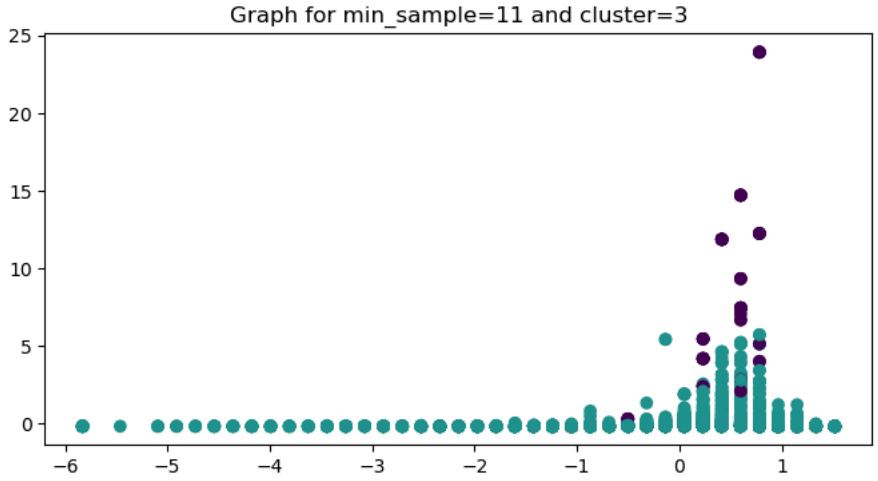



```
# min_samples = 11

dbscan=DBSCAN(eps=1,min_samples=11) # eps
dbscan.fit(X_scaled)
set(dbscan.labels_)
```

```
{-1, 0, 1}
```

```
plt.figure(figsize=(8,4))
plt.scatter(X_scaled[:,0],X_scaled[:,1],c=dbscan.labels_);
plt.title('Graph for min_sample=11 and cluster=3')
```

```
Text(0.5, 1.0, 'Graph for min_sample=11 and cluster=3')
```

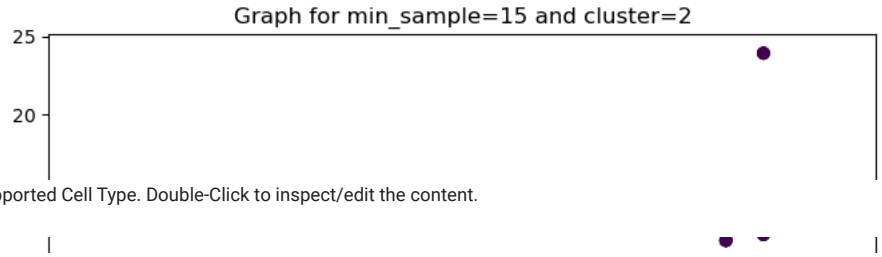



```
# min_samples = 15  (any value from 8 to 14 giving 3 clusters only)

dbscan=DBSCAN(eps=1,min_samples=15) # eps
dbscan.fit(X_scaled)
set(dbscan.labels_)
```

```
{-1, 0}
```

```
plt.figure(figsize=(8,4))
plt.scatter(X_scaled[:,0],X_scaled[:,1],c=dbscan.labels_);
plt.title('Graph for min_sample=15 and cluster=2')
```

```
Text(0.5, 1.0, 'Graph for min_sample=15 and cluster=2')
```



Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# creating DBSCAN model with 2 clusters.

dbscan = DBSCAN(eps=1, min_samples=15)
clusters = dbscan.fit_predict(X_scaled)

# Adding the cluster to the DataFrame
df_filtered = df.loc[X.index].copy()  # Filtering the original data frame to match the rows in X
df_filtered['Cluster'] = clusters

# Printing the resulting DataFrame with cluster labels
print(df_filtered[['App', 'Rating', 'Reviews', 'Size', 'Installs', 'Price', 'Cluster']].head())
```

```
                                                App  Rating Reviews  \
0     Photo Editor & Candy Camera & Grid & ScrapBook     4.1     159
1                               Coloring book moana     3.9     967
2  U Launcher Lite – FREE Live Cool Themes, Hide ...     4.7   87510
3                             Sketch - Draw & Paint     4.5  215644
4              Pixel Draw - Number Art Coloring Book     4.3     967

          Size    Installs  Price  Cluster
0  19000000.0     10000.0    0.0        0
1  14000000.0    500000.0    0.0        0
2   8700000.0   5000000.0    0.0        0
3  25000000.0  50000000.0    0.0        0
4   2800000.0    100000.0    0.0        0
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.