In [8]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as sch
import mpl_toolkits.mplot3d.axes3d as p3
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler, StandardScaler, RobustScaler
from sklearn import metrics, mixture, cluster, datasets
from sklearn.mixture import GaussianMixture
from sklearn.datasets import make_moons
from sklearn.cluster import AgglomerativeClustering, KMeans, DBSCAN
from sklearn.neighbors import kneighbors_graph
from itertools import cycle, islice
import time
from itertools import pairwise
from math import ceil
```

In [9]:
```
import warnings
warnings.filterwarnings('ignore')
sns.set_theme(context='notebook', palette='viridis')
```

In [10]:
```
pd.set_option('display.max_columns', 500)
```

In [11]:
```
df = pd.read_csv('6M-0K-99K.users.dataset.public.csv')
display(df.head(), df.describe(), df.info(), df.isnull().sum(), df.nunique())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98913 entries, 0 to 98912
Data columns (total 24 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   identifierHash       98913 non-null  int64  
 1   type                 98913 non-null  object 
 2   country              98913 non-null  object 
 3   language             98913 non-null  object 
 4   socialNbFollowers    98913 non-null  int64  
 5   socialNbFollows      98913 non-null  int64  
 6   socialProductsLiked  98913 non-null  int64  
 7   productsListed       98913 non-null  int64  
 8   productsSold         98913 non-null  int64  
 9   productsPassRate     98913 non-null  float64
 10  productsWished       98913 non-null  int64  
 11  productsBought       98913 non-null  int64  
 12  gender               98913 non-null  object 
 13  civilityGenderId     98913 non-null  int64  
 14  civilityTitle        98913 non-null  object 
 15  hasAnyApp            98913 non-null  bool   
 16  hasAndroidApp        98913 non-null  bool   
 17  hasIosApp            98913 non-null  bool   
 18  hasProfilePicture    98913 non-null  bool   
 19  daysSinceLastLogin   98913 non-null  int64  
 20  seniority            98913 non-null  int64  
 21  seniorityAsMonths    98913 non-null  float64
 22  seniorityAsYears     98913 non-null  float64
 23  countryCode          98913 non-null  object 
dtypes: bool(4), float64(3), int64(11), object(6)
memory usage: 15.5+ MB
```

	identifierHash	type	country	language	socialNbFollowers	socialNbFollows	socialProd
0	-1097895247965112460	user	Royaume-Uni	en	147	10	
1	2347567364561867620	user	Monaco	en	167	8	
2	6870940546848049750	user	France	fr	137	13	
3	-4640272621319568052	user	Etats-Unis	en	131	10	
4	-5175830994878542658	user	Etats-Unis	en	167	8	

	identifierHash	socialNbFollowers	socialNbFollows	socialProductsLiked	productsListed	prod
count	9.891300e+04	98913.000000	98913.000000	98913.000000	98913.000000	9891
mean	-6.692039e+15	3.432269	8.425677	4.420743	0.093304	
std	5.330807e+18	3.882383	52.839572	181.030569	2.050144	
min	-9.223101e+18	3.000000	0.000000	0.000000	0.000000	
25%	-4.622895e+18	3.000000	8.000000	0.000000	0.000000	
50%	-1.337989e+15	3.000000	8.000000	0.000000	0.000000	
75%	4.616388e+18	3.000000	8.000000	0.000000	0.000000	
max	9.223331e+18	744.000000	13764.000000	51671.000000	244.000000	17

```
None
identifierHash         0
type                   0
country                0
language               0
socialNbFollowers      0
socialNbFollows        0
socialProductsLiked    0
productsListed         0
productsSold           0
productsPassRate       0
productsWished         0
productsBought         0
gender                 0
civilityGenderId       0
civilityTitle          0
hasAnyApp              0
hasAndroidApp          0
hasIosApp              0
hasProfilePicture      0
daysSinceLastLogin     0
seniority              0
seniorityAsMonths      0
seniorityAsYears       0
countryCode            0
dtype: int64
```

```
identifierHash         98913
type                       1
country                  200
language                   5
socialNbFollowers         90
socialNbFollows           85
socialProductsLiked      420
productsListed            65
productsSold              75
productsPassRate          72
productsWished           279
productsBought            70
gender                     2
civilityGenderId           3
civilityTitle              3
hasAnyApp                  2
hasAndroidApp              2
hasIosApp                  2
hasProfilePicture          2
daysSinceLastLogin       699
seniority                 19
seniorityAsMonths         19
seniorityAsYears           6
countryCode              199
dtype: int64
```

In [12]:
```
df.shape
```

Out[12]:
```
(98913, 24)
```

In [13]:
```
df.columns
```

Out[13]:
```
Index(['identifierHash', 'type', 'country', 'language', 'socialNbFollowers',
       'socialNbFollows', 'socialProductsLiked', 'productsListed',
       'productsSold', 'productsPassRate', 'productsWished', 'productsBought',
       'gender', 'civilityGenderId', 'civilityTitle', 'hasAnyApp',
       'hasAndroidApp', 'hasIosApp', 'hasProfilePicture', 'daysSinceLastLogin',
       'seniority', 'seniorityAsMonths', 'seniorityAsYears', 'countryCode'],
      dtype='object')
```

In [14]:
```
df.drop(['identifierHash'], axis=1, inplace=True)
```

In [15]:
```
# check for duplicates
df.duplicated().sum()
```

Out[15]:
```
49153
```

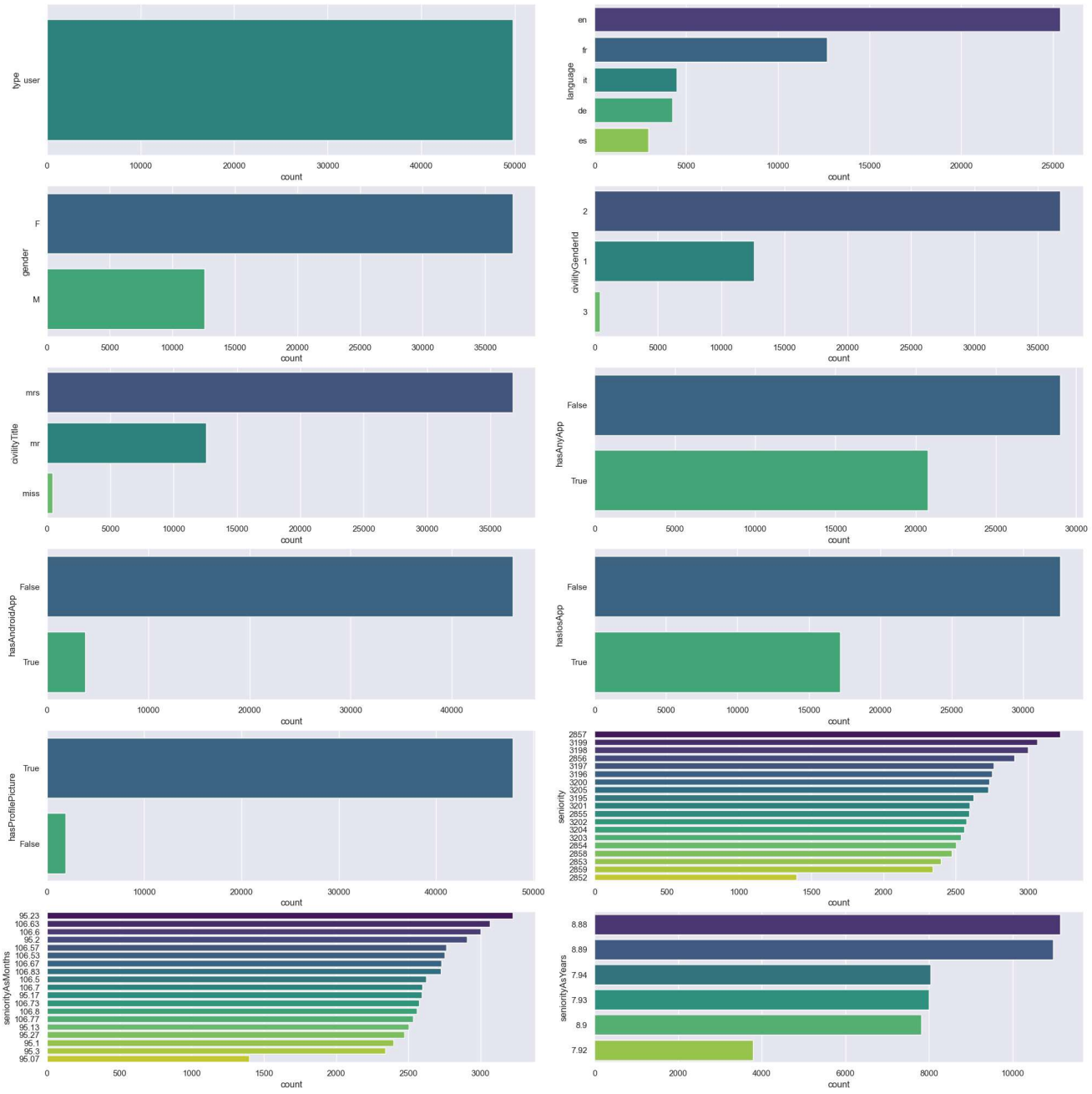
In [16]:
```
df.drop_duplicates(inplace=True)
```

In [17]:
```
df.shape
```

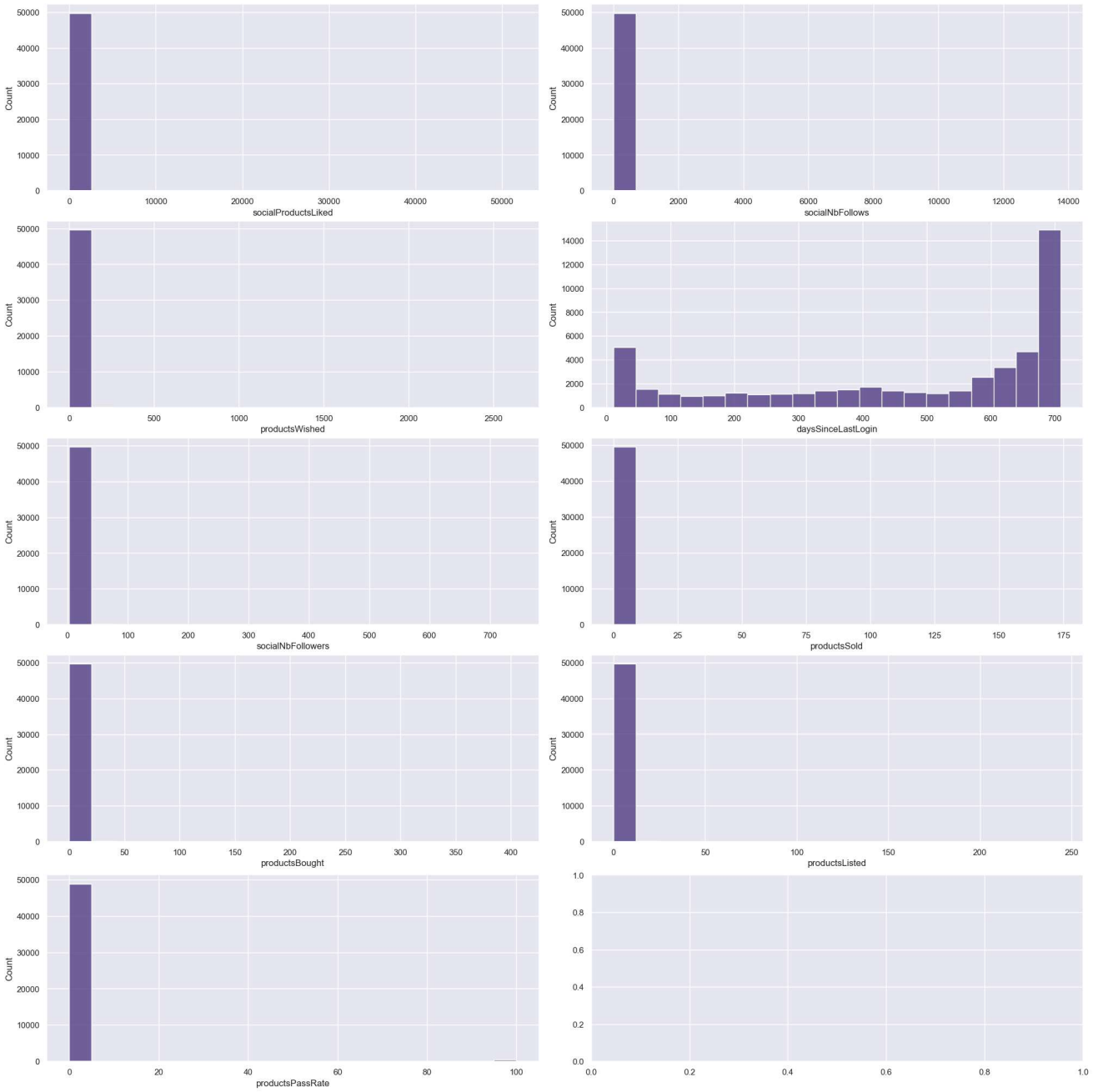
Out[17]:
```
(49760, 23)
```

In [18]:
```
fig, axs = plt.subplots(6, 2, figsize=(20, 20), constrained_layout=True)
categorical = ['type', 'language', 'gender',
               'civilityGenderId', 'civilityTitle', 'hasAnyApp', 'hasAndroidApp',
               'hasIosApp', 'hasProfilePicture', 'seniority',
               'seniorityAsMonths', 'seniorityAsYears', ]
for i, f in enumerate(categorical):
```

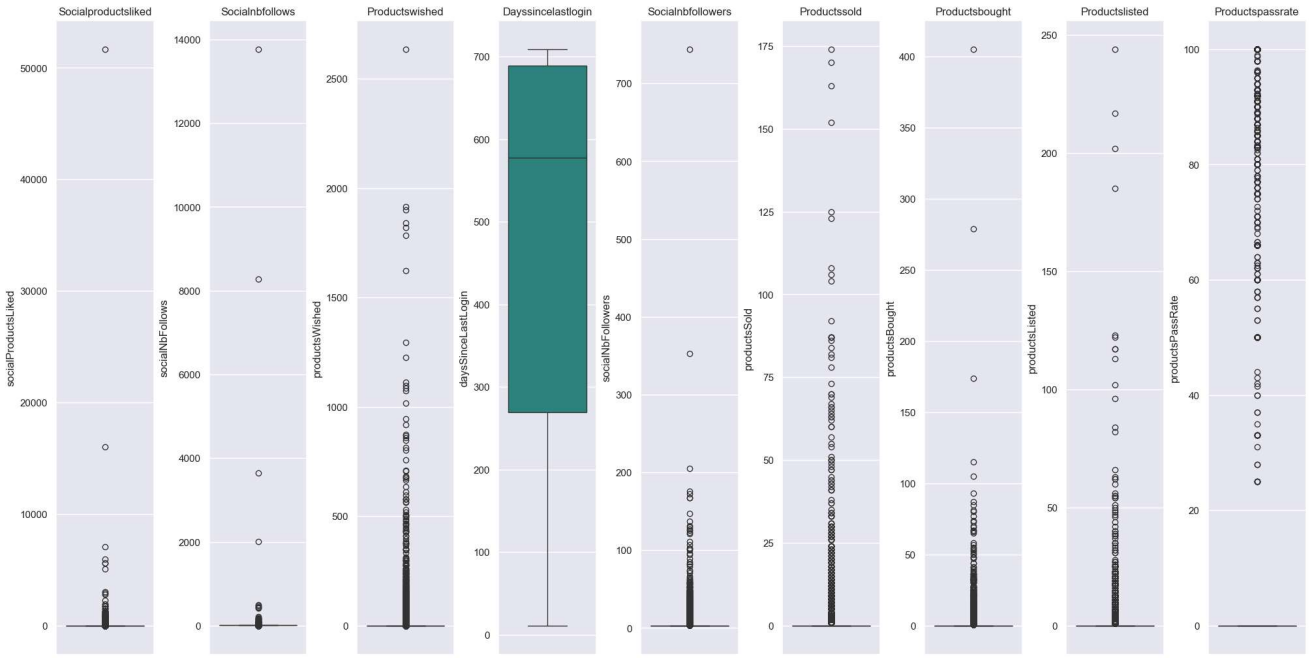
```
    sns.countplot(y=f, data=df,
                  ax=axs[i//2][i % 2],
                  order=df[f].value_counts().index, palette='viridis')
```



In [19]:
```
numerical = set(df.columns)-set(categorical)-{'countryCode', 'country'}
fig, axs = plt.subplots(len(numerical)//2+1, 2,
                        figsize=(20, 20), constrained_layout=True)

for i, f in enumerate(numerical):
    sns.histplot(x=f, data=df,
                 ax=axs[i//2][i % 2], bins=20,
                 palette=sns.color_palette("viridis", 2))

plt.show()
```



In [20]:
```
fig, axs = plt.subplots(1, 9,
                        figsize=(20, 10), constrained_layout=True)
for i, f in enumerate(numerical):
    sns.boxplot(y=f,  data=df,  ax=axs[i],
                palette="viridis")
    axs[i].set_title(f.capitalize())


plt.show()
```



In [21]:
```
numerical
```

Out[21]:
```
{'daysSinceLastLogin',
 'productsBought',
 'productsListed',
 'productsPassRate',
 'productsSold',
 'productsWished',
 'socialNbFollowers',
 'socialNbFollows',
 'socialProductsLiked'}
```

In [22]:
```
def check_counts(df, col, int1, int2):
    bin1_count = df[df[col] <= int1].shape[0]
    bin2_count = df[(df[col] > int1) & (df[col] < int2)].shape[0]
    bin3_count = df[df[col] >= int2].shape[0]
    print(f'{col} bin sizes: {bin1_count, bin2_count, bin3_count}')
```

In [23]:
```
check_counts(df, 'socialProductsLiked', 0, 5)
check_counts(df, 'socialNbFollows', 8, 10)
check_counts(df, 'socialNbFollowers', 3, 5)
check_counts(df, 'productsBought', 0, 5)
check_counts(df, 'productsWished', 0, 5)
check_counts(df, 'productsSold', 0, 5)
check_counts(df, 'productsListed', 0, 3)
check_counts(df, 'productsPassRate', 0, 80)
```

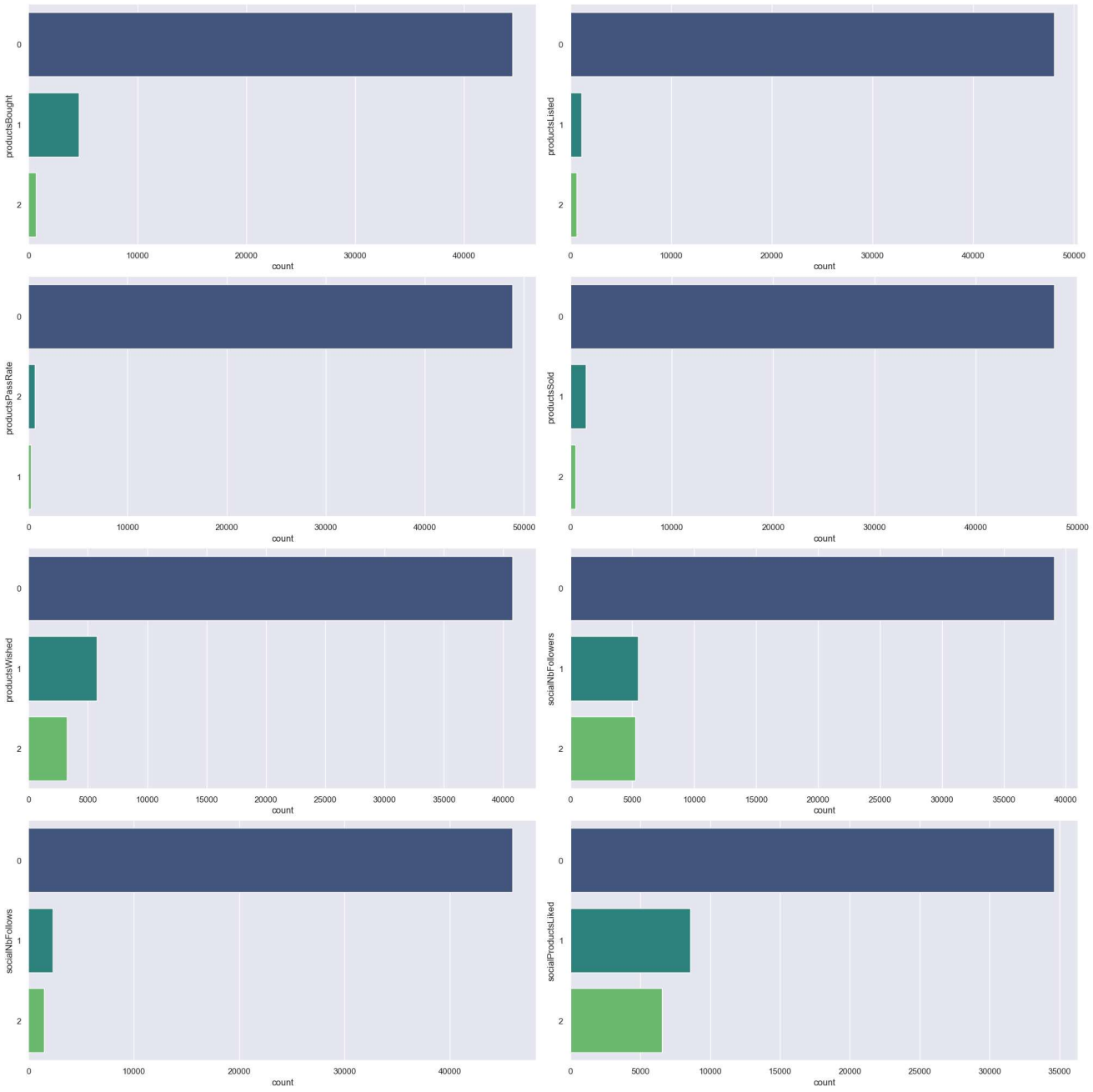
```
socialProductsLiked bin sizes: (34614, 8580, 6566)
socialNbFollows bin sizes: (45936, 2329, 1495)
socialNbFollowers bin sizes: (39059, 5474, 5227)
productsBought bin sizes: (44428, 4633, 699)
productsWished bin sizes: (40753, 5747, 3260)
productsSold bin sizes: (47724, 1520, 516)
productsListed bin sizes: (48036, 1086, 638)
productsPassRate bin sizes: (48826, 266, 668)
```

In [24]:
```
def discretization(df, column, val1, val2):
    conditions = [
        (df[column] <= val1),
        (df[column] > val1) & (df[column] < val2),
```

```
        (df[column] >= val2)
    ]

    values = [0, 1, 2]
    df[column] = np.select(conditions, values, default=0)
```

In [25]:
```
discretization(df, 'socialProductsLiked', 0, 5)
discretization(df, 'socialNbFollows', 8, 10)
discretization(df, 'socialNbFollowers', 3, 5)
discretization(df, 'productsBought', 0, 5)
discretization(df, 'productsWished', 0, 5)
discretization(df, 'productsSold', 0, 5)
discretization(df, 'productsListed', 0, 3)
discretization(df, 'productsPassRate', 0, 80)
```

In [26]:
```
fig, axs = plt.subplots(4, 2, figsize=(20, 20), constrained_layout=True)
cols = ['productsBought', 'productsListed', 'productsPassRate',
        'productsSold', 'productsWished', 'socialNbFollowers', 'socialNbFollows', 'social
for i, f in enumerate(cols):
    sns.countplot(y=f, data=df,
                  ax=axs[i//2][i % 2],
                  order=df[f].value_counts().index, palette='viridis')
```



In [27]:
```
df.head()
```

Out[27]:

	type	country	language	socialNbFollowers	socialNbFollows	socialProductsLiked	productsLi
0	user	Royaume-Uni	en	2	2	2	
1	user	Monaco	en	2	0	1	
2	user	France	fr	2	2	2	
3	user	Etats-Unis	en	2	2	2	
4	user	Etats-Unis	en	2	0	0	

In [28]:
```
df.drop(['type'], axis=1, inplace=True)
```

In [29]:
```
df.head()
```

Out[29]:

	country	language	socialNbFollowers	socialNbFollows	socialProductsLiked	productsListed
0	Royaume-Uni	en	2	2	2	2
1	Monaco	en	2	0	1	2
2	France	fr	2	2	2	2
3	Etats-Unis	en	2	2	2	2
4	Etats-Unis	en	2	0	0	2

In [30]:
```
# encode categorical
df["gender"] = df["gender"].map({'M': 0, 'F': 1})
```

In [31]:
```
country_count = dict(df['countryCode'].value_counts())
df['countryCode'] = df['countryCode'].map(country_count)
```

In [32]:
```
df.head()
```

Out[32]:

	country	language	socialNbFollowers	socialNbFollows	socialProductsLiked	productsListed
0	Royaume-Uni	en	2	2	2	2
1	Monaco	en	2	0	1	2
2	France	fr	2	2	2	2
3	Etats-Unis	en	2	2	2	2
4	Etats-Unis	en	2	0	0	2

In [33]:
```
country_count = dict(df['country'].value_counts())
df['country'] = df['country'].map(country_count)
```

In [34]:
```
df = pd.get_dummies(df, columns=['language', 'civilityTitle'])
```

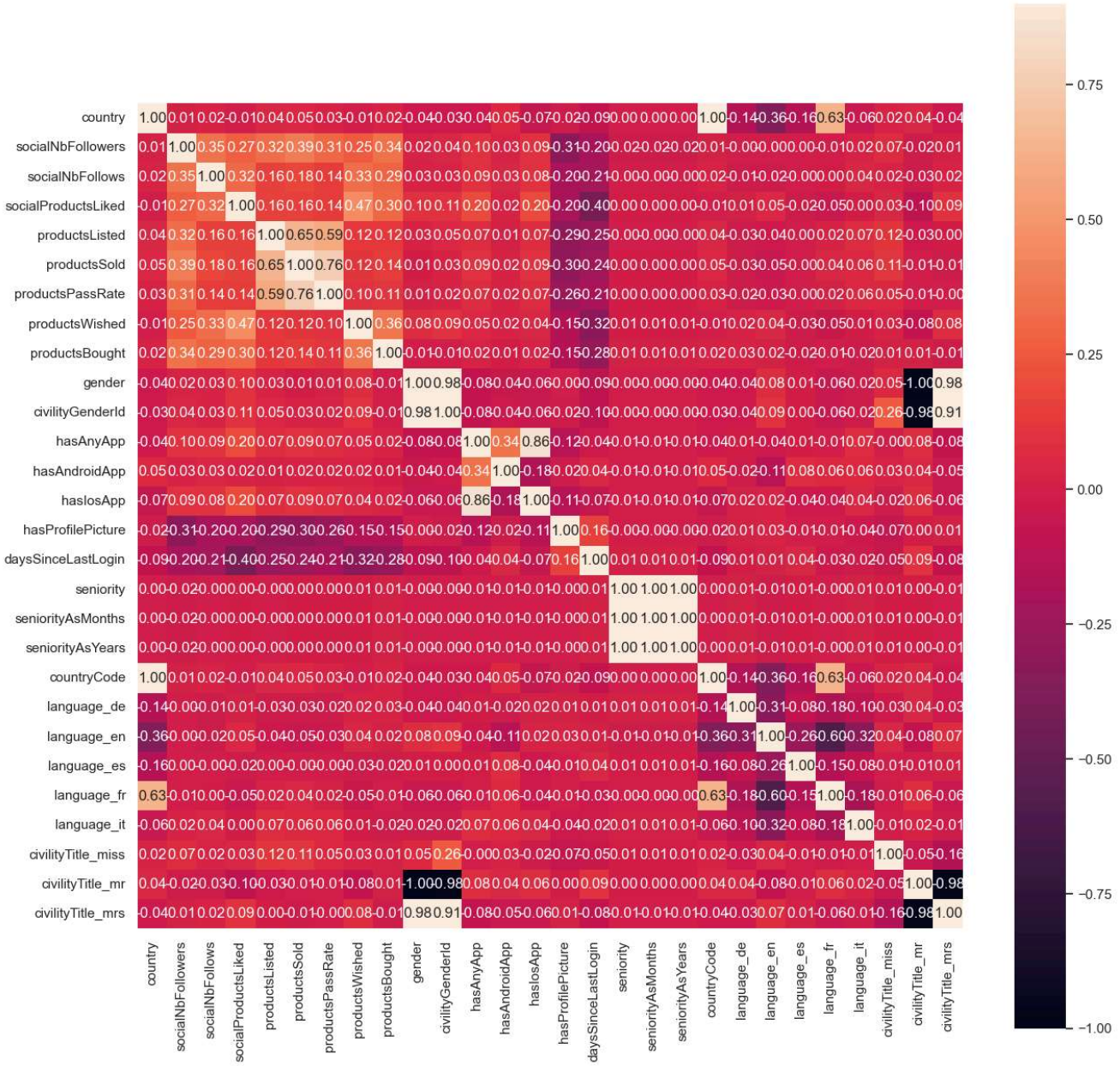
In [35]:
```
df.head()
```

Out[35]:

	country	socialNbFollowers	socialNbFollows	socialProductsLiked	productsListed	productsSold
0	5454	2	2	2	2	2
1	34	2	0	1	2	2
2	11145	2	2	2	2	2
3	7459	2	2	2	2	2
4	7459	2	0	0	2	2

In [36]:
```
corr_matrix = df.corr()
plt.figure(figsize=(15, 15))
sns.heatmap(corr_matrix,  vmax=.9, square=True,
            annot=True, fmt='.2f')
```

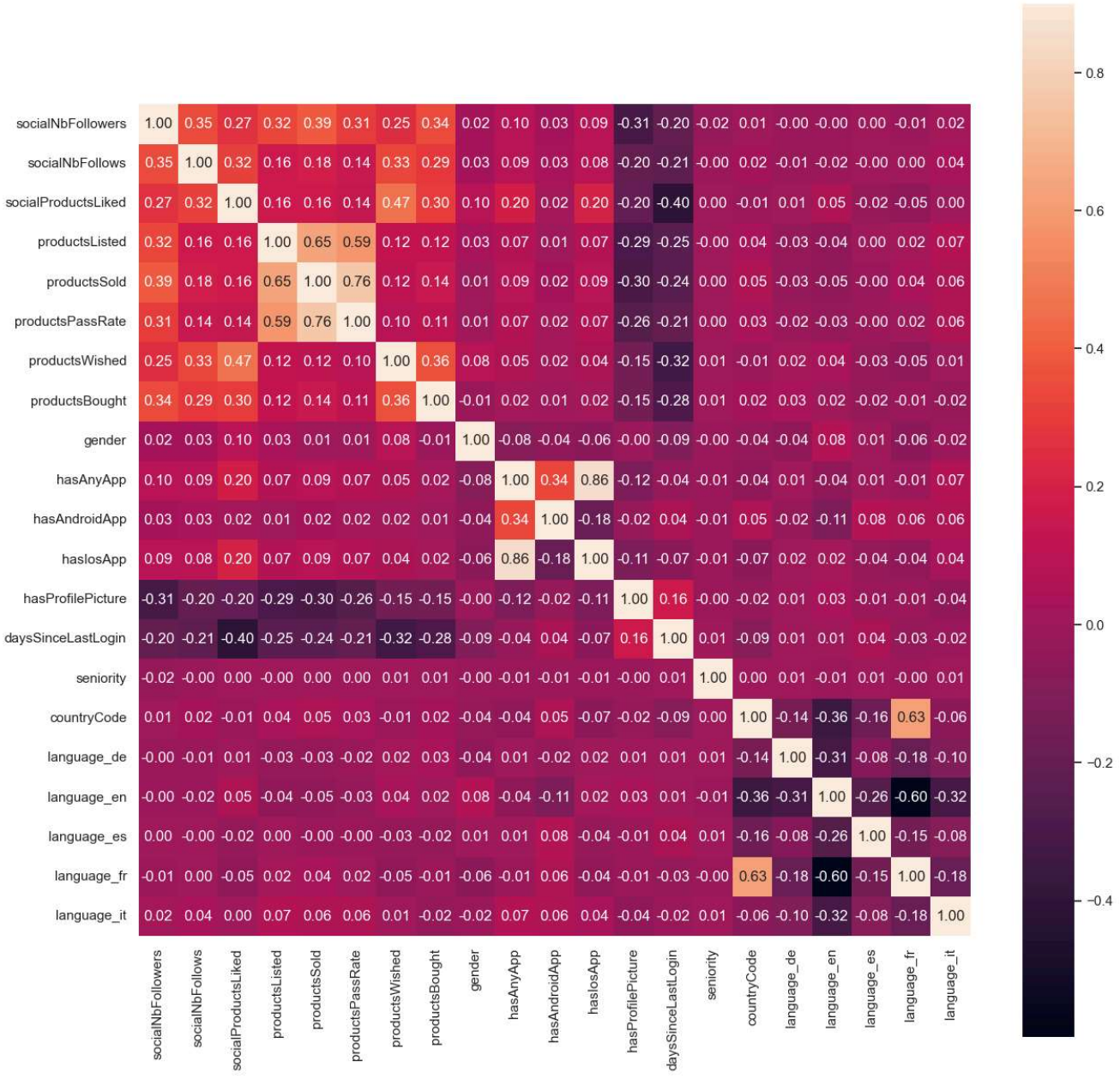
Out[36]: <Axes: >



In [37]:
```
# drop highly correlated features
df.drop(['civilityTitle_miss', 'civilityTitle_mr', 'civilityTitle_mrs', 'civilityGenderId
         'seniorityAsMonths', 'seniorityAsYears', 'country'], axis=1, inplace=True)
```

In [38]:
```
corr_matrix = df.corr()
plt.figure(figsize=(15, 15))
sns.heatmap(corr_matrix,  vmax=.9, square=True,
            annot=True, fmt='.2f')
```

Out[38]: <Axes: >



In [33]:
```
scaler = MinMaxScaler((-1, 1))
df = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)

df.describe()
```

Out[33]:

	socialNbFollowers	socialNbFollows	socialProductsLiked	productsListed	productsSold	pro
count	49760.000000	49760.000000	49760.000000	49760.000000	49760.000000	
mean	-0.679904	-0.893107	-0.563666	-0.952532	-0.948714	
std	0.654012	0.394409	0.714047	0.266194	0.263433	
min	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	
25%	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	
50%	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	
75%	-1.000000	-1.000000	0.000000	-1.000000	-1.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	

In [34]:
```
# from sklearn.neighbors import LocalOutlierFactor

# clf = LocalOutlierFactor(n_neighbors=20, contamination=0.05)
# y_pred = clf.fit_predict(df)
# df = df[y_pred == 1]

# df.shape
```

In [35]:
```
from sklearn.ensemble import IsolationForest

clf = IsolationForest(contamination=0.03)
y_pred = clf.fit_predict(df)
df = df[y_pred == 1]

df.shape
```

Out[35]: (48267, 21)

In [36]:
```
df.head()
```

Out[36]:

	socialNbFollowers	socialNbFollows	socialProductsLiked	productsListed	productsSold	produ
11	1.0	-1.0	-1.0	-1.0	1.0	
78	1.0	-1.0	-1.0	-1.0	1.0	
130	1.0	-1.0	-1.0	0.0	1.0	
187	1.0	-1.0	-1.0	-1.0	1.0	
245	1.0	-1.0	0.0	-1.0	1.0	

In [37]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import plotly.express as px
```

```
def plot_3d(X_pca, labels, title='3D PCA Cluster Plot'):
    fig = px.scatter_3d(x=X_pca[:, 0], y=X_pca[:, 1], z=X_pca[:, 2], color=labels,
                        labels={'x': 'PCA 0',
                                'y': 'PCA 1', 'z': 'PCA 2'},
                        title=title)
    fig.update_traces(marker=dict(size=3))
    fig.show()
```

In [38]:
```
def plot_2d(X_pca, clusters):
    plt.figure(figsize=(20, 5))
    plots_components = [(0, 1), (0, 2), (0, 3), (0, 4)]

    for i, (x, y) in enumerate(plots_components):
        plt.subplot(1, 4, i+1)
        plt.scatter(X_pca[:, x], X_pca[:, y], c=clusters,
                    cmap='viridis', s=50)
        plt.title(f'Original Data with True Clusters {x} vs {y}')
        plt.xlabel(f'Principal Component {x}')
        plt.ylabel(f'Principal Component {y}')
    plt.show()
```

In [39]:
```
def plot_feature_importance(df, clusters, centroids=None):
    if centroids is None:
        centroids = df.groupby(clusters).mean()

    feature_importance = centroids.std(axis=0)
    plt.figure(figsize=(10, 5))
    plt.bar(df.columns, feature_importance)
    plt.xticks(rotation=90)
    plt.title('Feature importance based on centroids')
    plt.show()

    return feature_importance
```

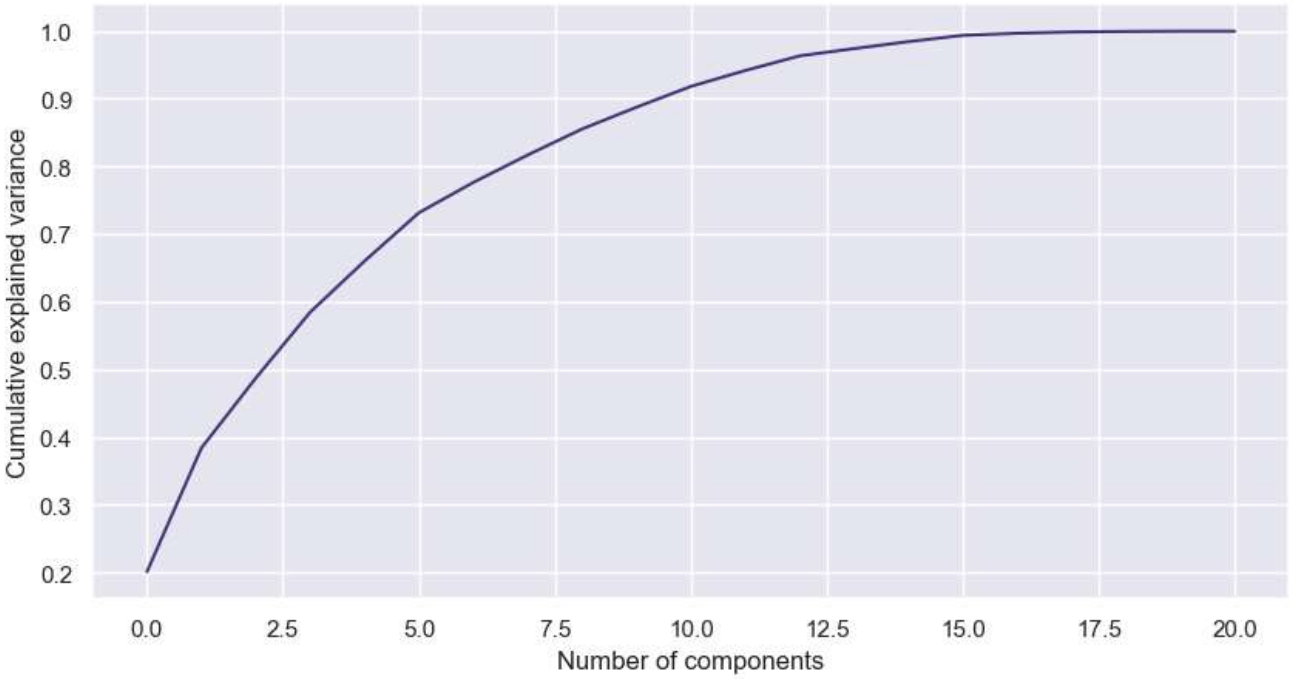
In [40]:
```
def analyze_features(df, feature_importance, clusters, num_clusters, thershold=0.25):
    important_columns = df.columns[feature_importance > thershold]
    cluster_data = np.array(
        [df[clusters == i][important_columns].mean(axis=0) for i in range(num_clusters)])
    cluster_data = pd.DataFrame(
        cluster_data.T, columns=range(num_clusters), index=important_columns)

    return cluster_data
```

In [41]:
```
results = pd.DataFrame(
    columns=['Model', 'Silhouette Score', 'Calinski-Harabasz Score'])
results.set_index('Model', inplace=True)
```

In [42]:
```
def register_results(results, model, X, clusters):
    results.loc[model, 'Silhouette Score'] = metrics.silhouette_score(
        X, clusters, sample_size=10000)
    results.loc[model,
                'Calinski-Harabasz Score'] = metrics.calinski_harabasz_score(X, clusters)
    display(results.loc[[model]])
```

In [43]:
```
# apply pca
from sklearn.decomposition import PCA
pca = PCA()
pca.fit(df)
```

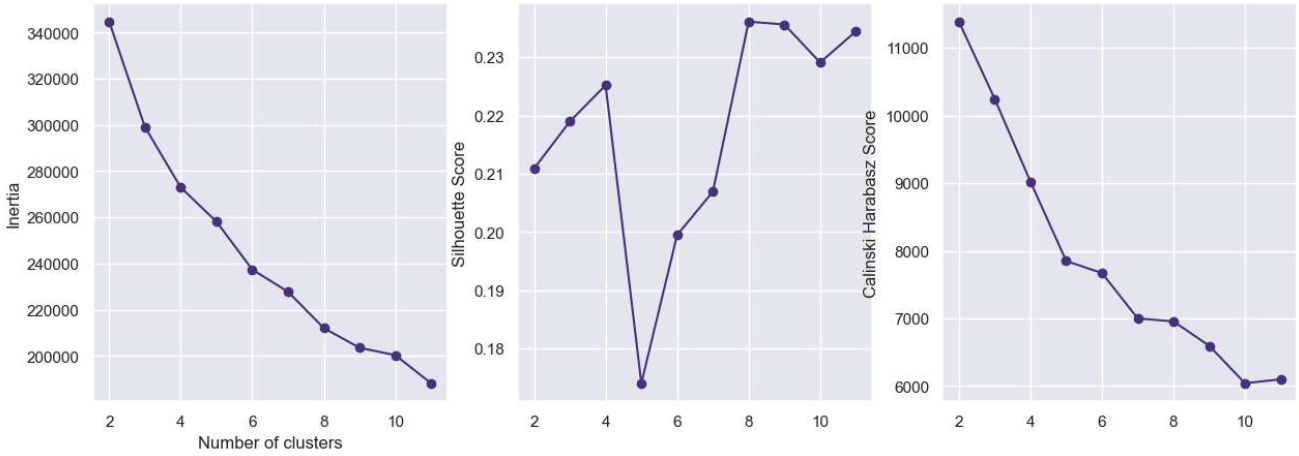
```
data_pca = pca.transform(df)

# plot pca
plt.figure(figsize=(10, 5))
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of components')
plt.ylabel('Cumulative explained variance')
plt.show()
```



K-Means

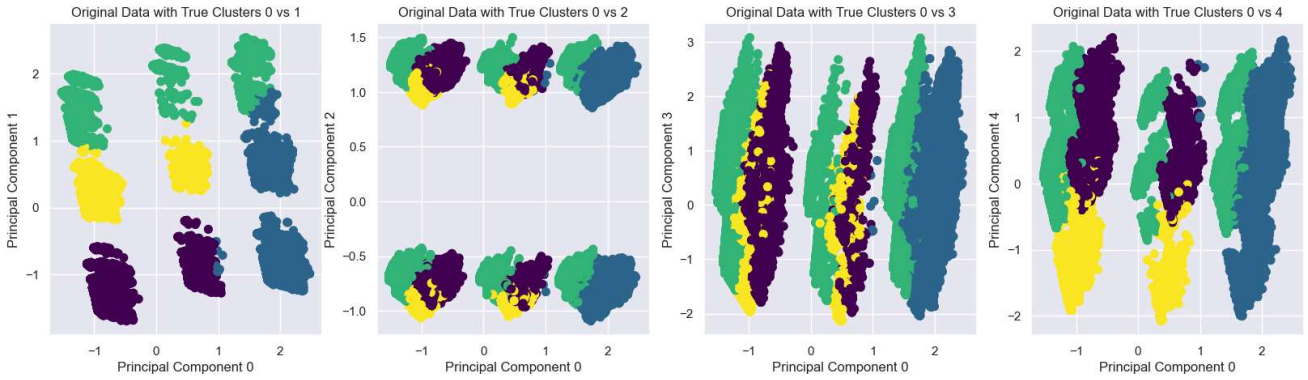
In [42]:
```
k_range = range(2, 12)
inertia = []
silhouette = []
calinski = []
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df)
    inertia.append(kmeans.inertia_)
    silhouette.append(metrics.silhouette_score(
        df, kmeans.labels_, sample_size=10000))
    calinski.append(metrics.calinski_harabasz_score(df, kmeans.labels_))
    print(k, end=' ')

plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.plot(k_range, inertia, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.subplot(1, 3, 2)
plt.plot(k_range, silhouette, marker='o')
plt.ylabel('Silhouette Score')
plt.subplot(1, 3, 3)
plt.plot(k_range, calinski, marker='o')
plt.ylabel('Calinski Harabasz Score')
plt.show()
```

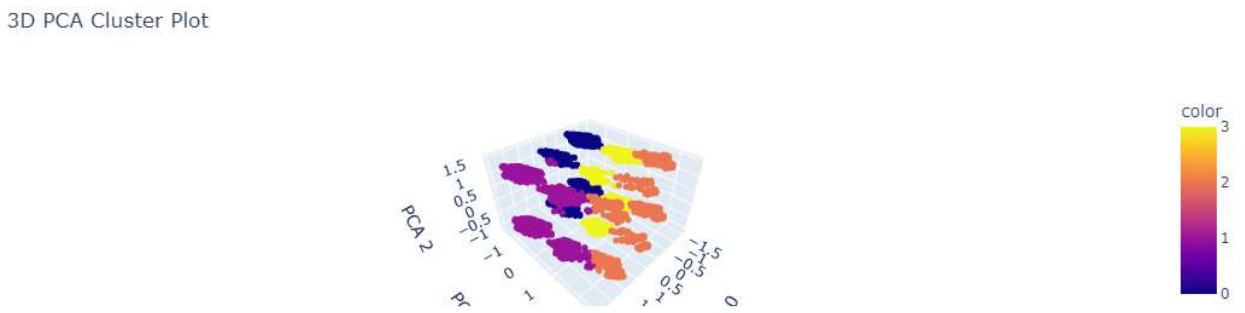
```
2 3 4 5 6 7 8 9 10 11
```



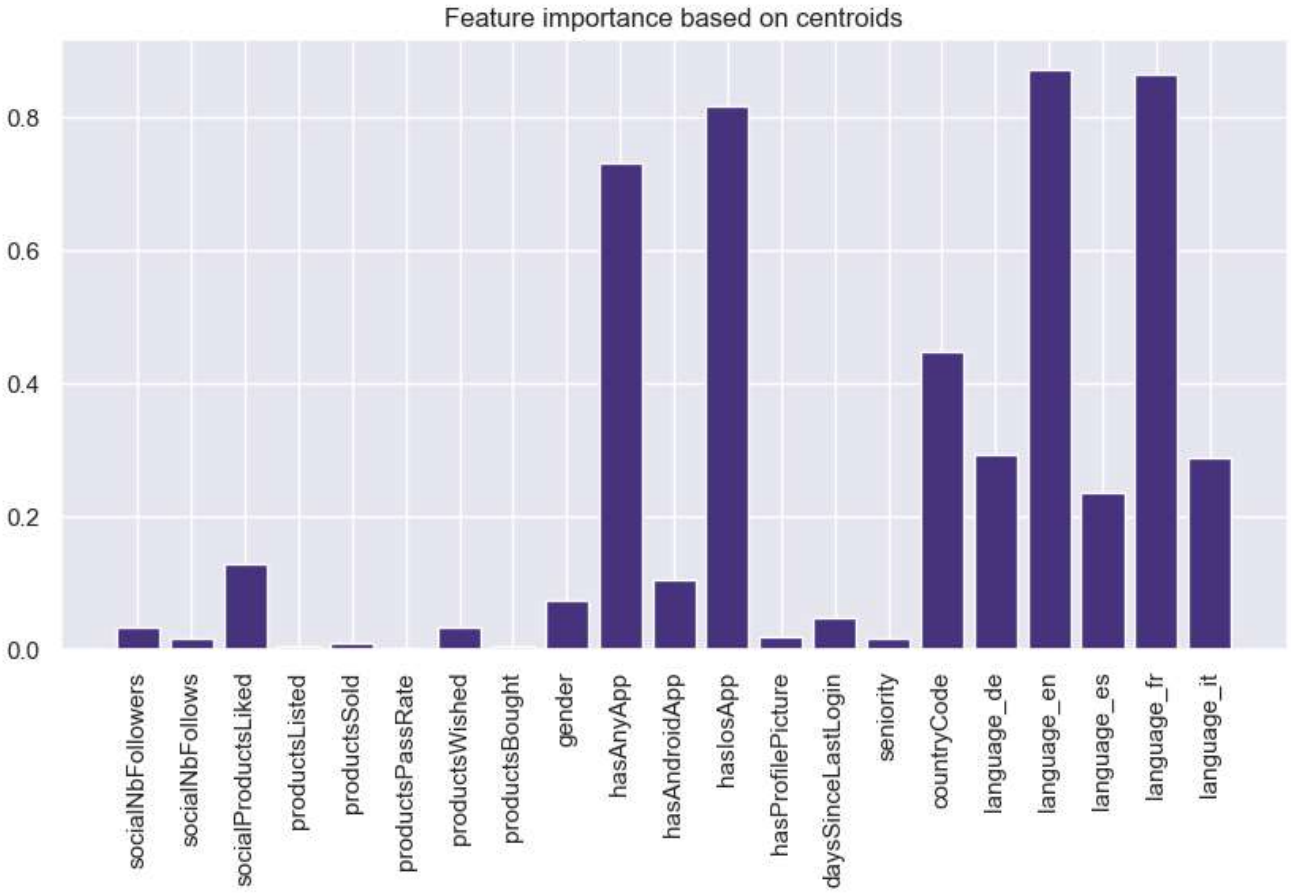
In [46]:
```
pca = PCA(n_components=5)
X_pca = pca.fit_transform(df)

NUM_CLUSTERS = 4
kmeans = KMeans(n_clusters=NUM_CLUSTERS, random_state=42)
clusters = kmeans.fit_predict(df)
register_results(results, 'KMeans(K=4)', df, clusters)

plot_2d(X_pca, clusters)
plot_3d(X_pca, clusters)
```

Model	Silhouette Score	Calinski-Harabasz Score
KMeans(K=4)	0.224022	9022.603463





In [47]:
```
feature_importance =\
    plot_feature_importance(df, clusters, centroids=kmeans.cluster_centers_)
analyze_features(df, feature_importance, clusters, NUM_CLUSTERS)
```



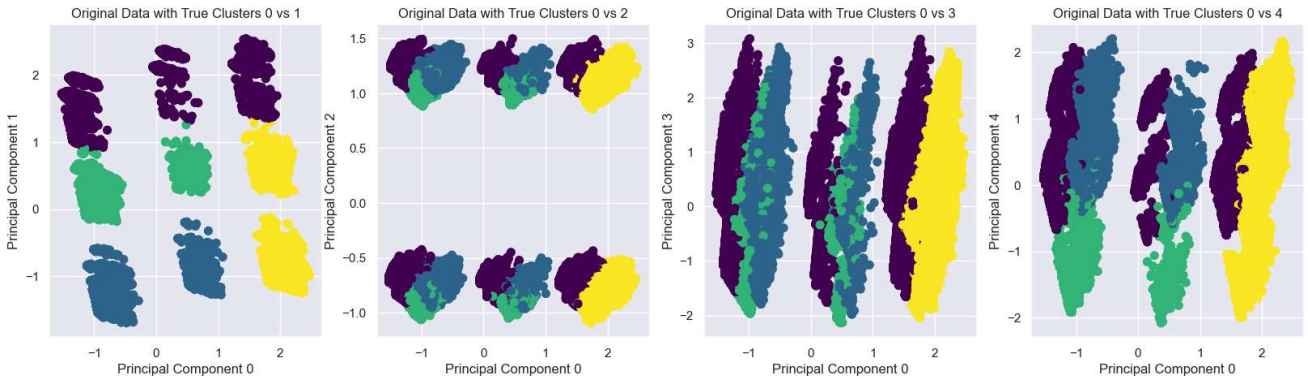
Out[47]:

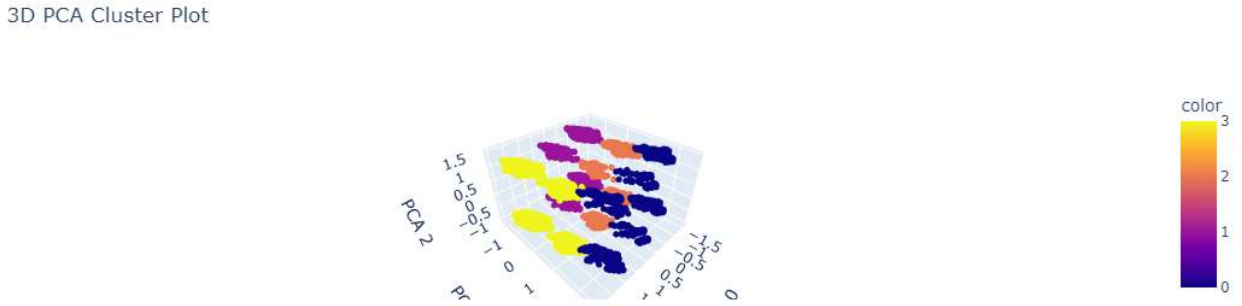
	0	1	2	3
hasAnyApp	-0.871831	1.000000	-0.210384	-0.694013
hasIosApp	-1.000000	0.998895	-0.401462	-1.000000
countryCode	-0.243314	-0.385427	0.708393	-0.331281
language_de	-1.000000	-0.762969	-1.000000	-0.279379
language_en	1.000000	0.374023	-1.000000	-1.000000
language_fr	-1.000000	-0.981366	1.000000	-1.000000
language_it	-1.000000	-0.747651	-1.000000	-0.294900

In [48]:
```
nonimportant_columns = df.columns[feature_importance < 0.2]
df_features = df.drop(columns=nonimportant_columns)
```

In [49]:
```
pca = PCA(n_components=5)
X_pca = pca.fit_transform(df)

NUM_CLUSTERS = 4
kmeans = KMeans(n_clusters=NUM_CLUSTERS, random_state=42)
clusters = kmeans.fit_predict(df_features)
register_results(results, 'KMeans(K=4,important cols)', df_features, clusters)

plot_2d(X_pca, clusters)
plot_3d(X_pca, clusters)
```

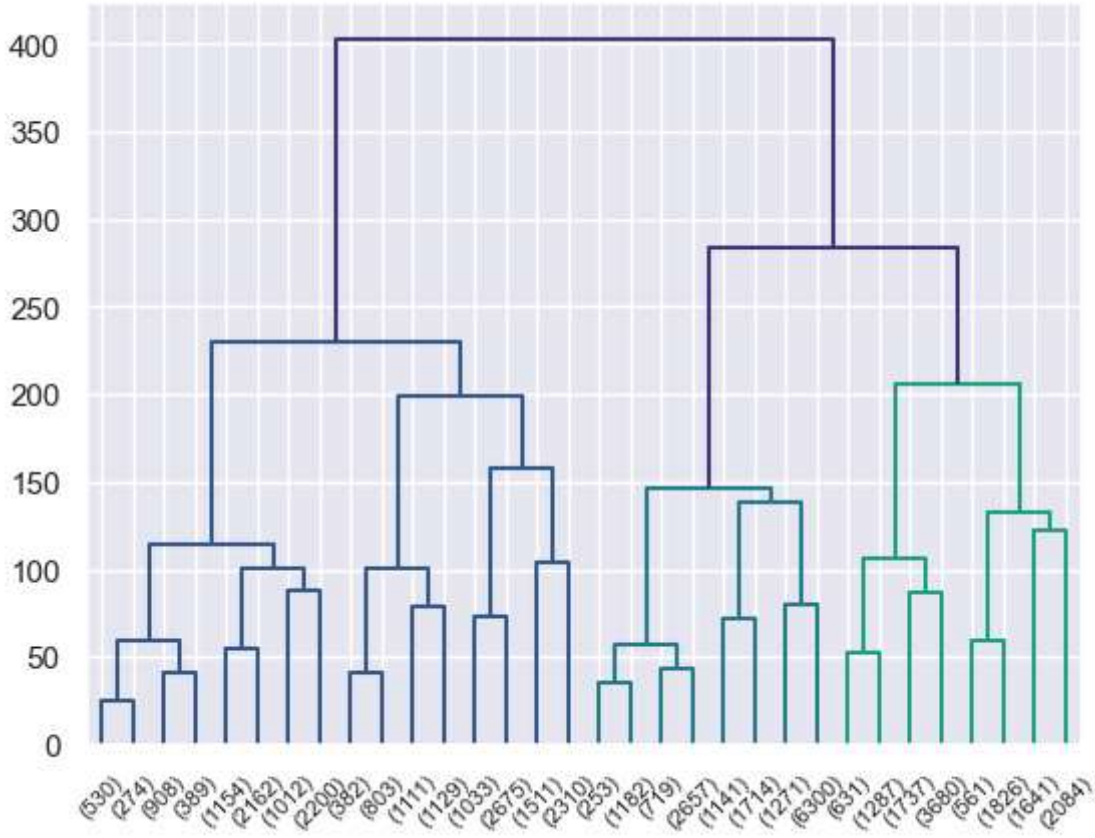
Model	Silhouette Score	Calinski-Harabasz Score
KMeans(K=4,important cols)	0.520517	27729.828964





Hijerarhijska klasterizacija i analiza dendrograma

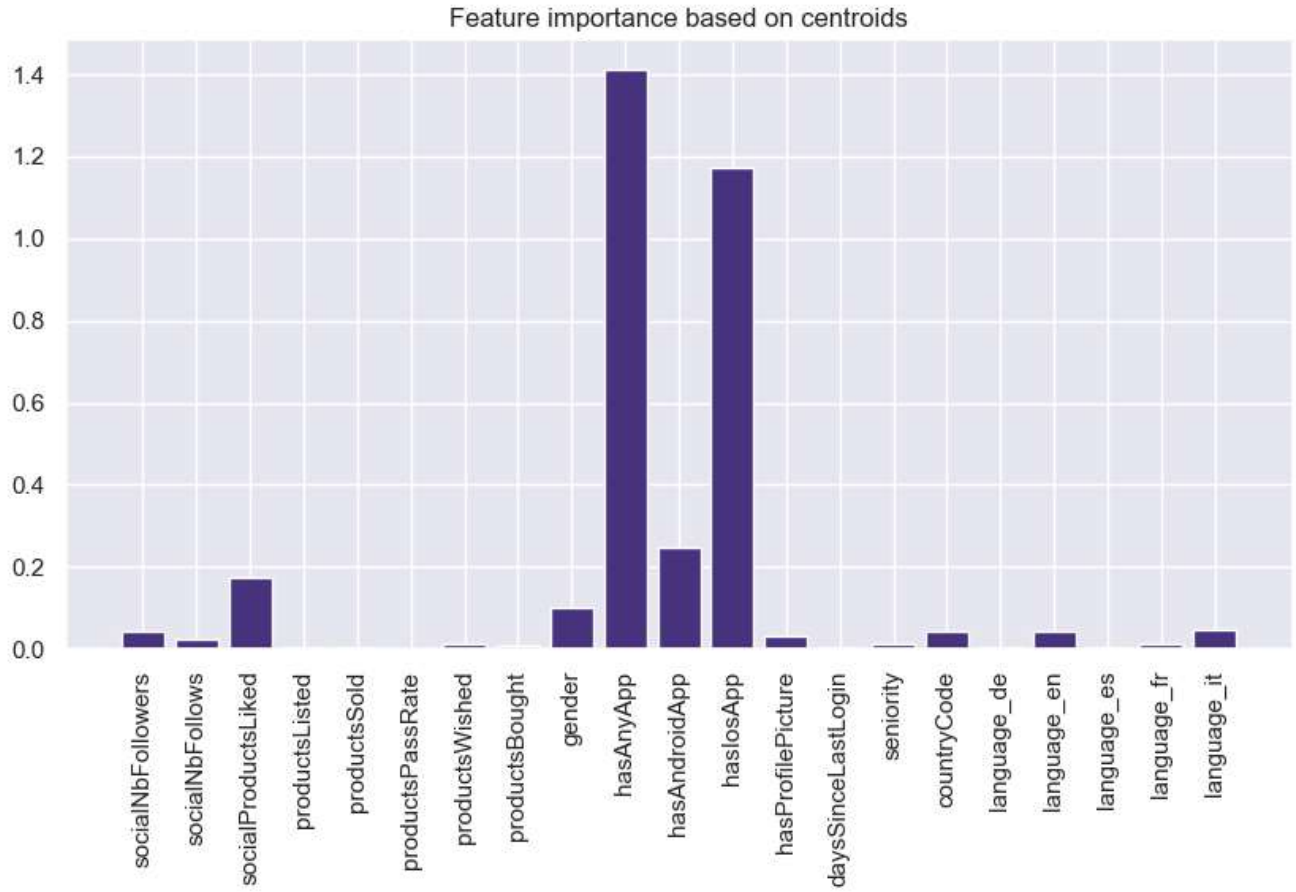
In [50]:
```
X = df.values
linkage_matrix = sch.linkage(X, method='ward')

dendrogram = sch.dendrogram(linkage_matrix, truncate_mode='level', p=4)
```



In [51]:
```
for cut_height in [350, 250, 220, 175]:
    clusters = sch.fcluster(linkage_matrix, cut_height, criterion='distance')
    plot_3d(X_pca, clusters, f'Cut height: {cut_height}')
    register_results(
        results, f'Hierarchical (dendrogram cut height = {cut_height})', df, clusters)
    fi = plot_feature_importance(df, clusters-1)
    display(analyze_features(df, fi, clusters-1, clusters.max()))
```

Cut height: 350

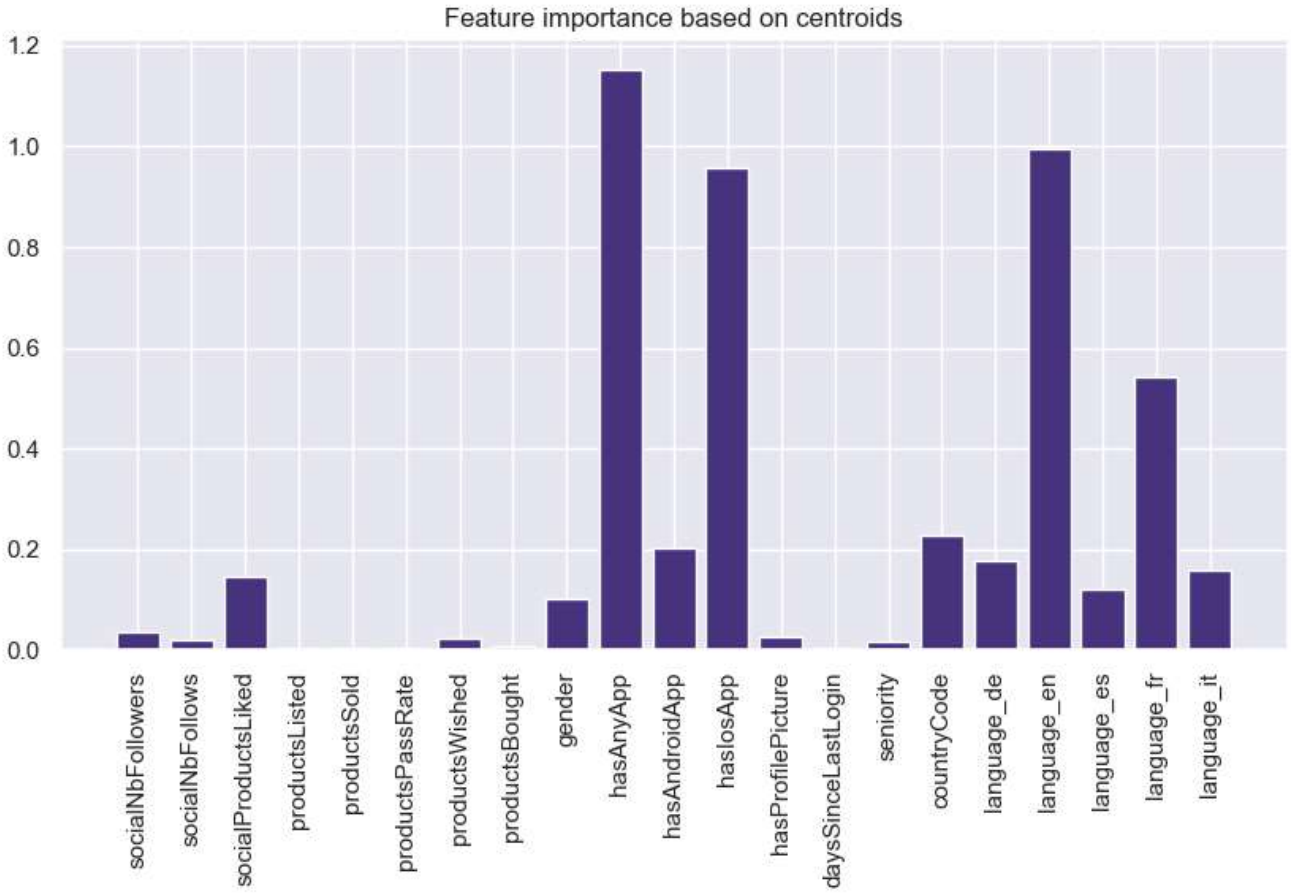
Model	Silhouette Score	Calinski-Harabasz Score
Hierarchical (dendrogram cut height = 350)	0.209615	11347.575582



	0	1
hasAnyApp	1.000000	-0.997839
hasIosApp	0.661645	-0.997839

Cut height: 250

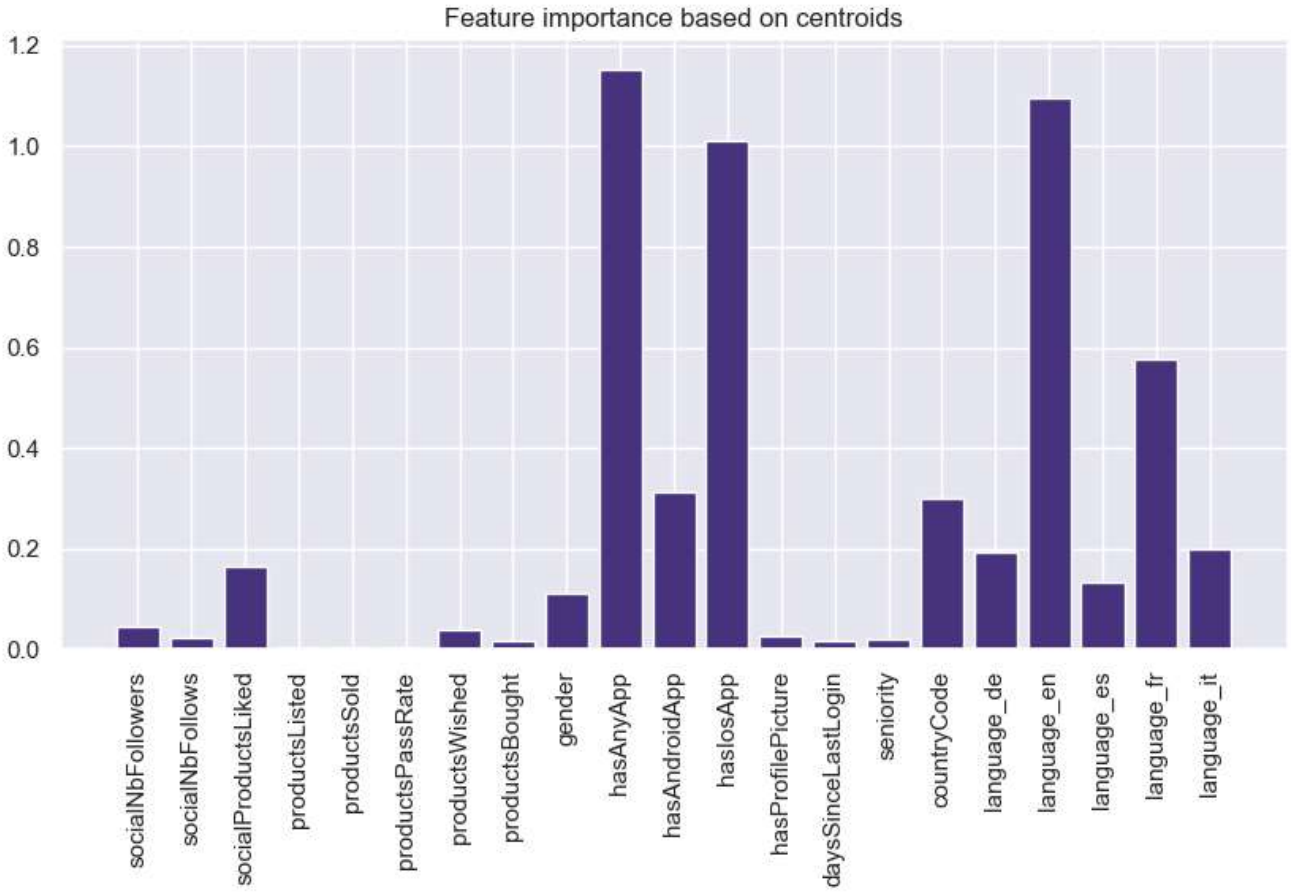
Model	Silhouette Score	Calinski-Harabasz Score
Hierarchical (dendrogram cut height = 250)	0.176792	9618.795831



	0	1	2
hasAnyApp	1.000000	-0.995931	-1.00000
hasIosApp	0.661645	-0.995931	-1.00000
language_en	-0.005260	0.991731	-1.00000
language_fr	-0.500281	-0.991731	0.09095

Cut height: 220

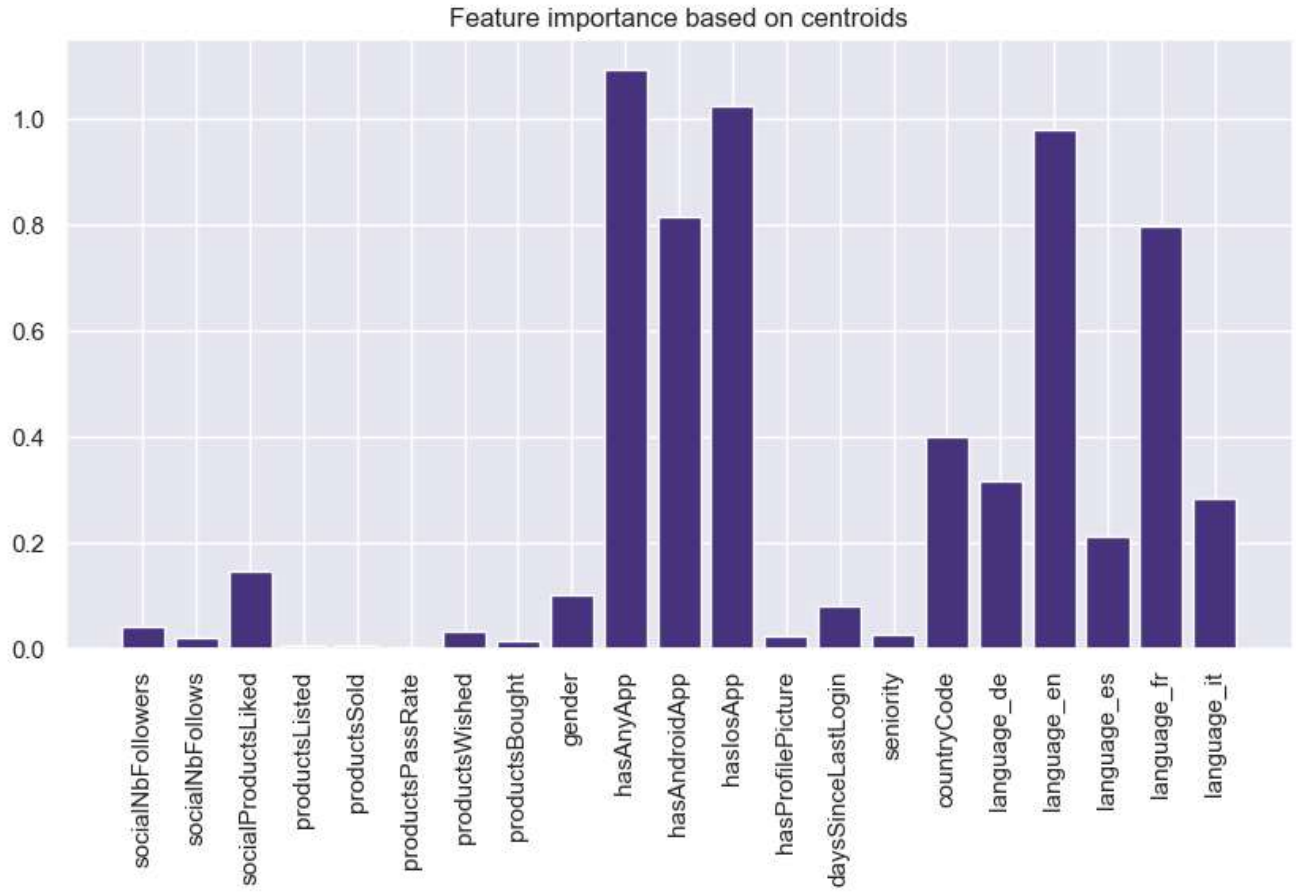
Model	Silhouette Score	Calinski-Harabasz Score
Hierarchical (dendrogram cut height = 220)	0.197746	8538.205959



	0	1	2	3
hasAnyApp	1.000000	1.000000	-0.995931	-1.000000
hasAndroidApp	-0.999073	-0.373562	-1.000000	-1.000000
hasIosApp	1.000000	0.395107	-0.995931	-1.000000
countryCode	-0.403196	0.158602	-0.240381	0.209043
language_en	1.000000	-0.797152	0.991731	-1.000000
language_fr	-1.000000	-0.106628	-0.991731	0.090950

Cut height: 175

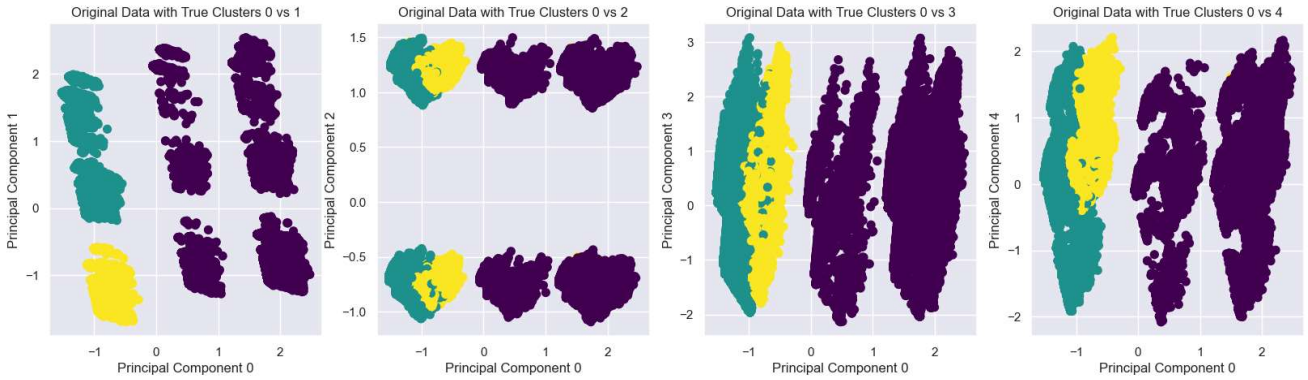
Model	Silhouette Score	Calinski-Harabasz Score
Hierarchical (dendrogram cut height = 175)	0.248036	7675.246798



	0	1	2	3	4	5
hasAnyApp	1.000000	1.000000	1.000000	-0.995931	-1.000000	-1.000000
hasAndroidApp	-0.999073	1.000000	-0.998406	-1.000000	-1.000000	-1.000000
hasIosApp	1.000000	-0.934599	1.000000	-0.995931	-1.000000	-1.000000
countryCode	-0.403196	0.087199	0.191083	-0.240381	0.657396	-0.329025
language_de	-1.000000	-0.871533	-0.603135	-1.000000	-1.000000	-0.218914
language_en	1.000000	-0.351241	-1.000000	0.991731	-1.000000	-1.000000
language_fr	-1.000000	-0.308029	-0.015009	-0.991731	1.000000	-1.000000
language_it	-1.000000	-0.723212	-0.579227	-1.000000	-1.000000	-0.318063

Agglomerative

In [52]:
```
agg_clustering = AgglomerativeClustering(n_clusters=3)
labels = agg_clustering.fit_predict(df)

plot_2d(X_pca, labels)
plot_3d(X_pca, labels)
register_results(results, 'Agglomerative (K=3)', df, labels)
```

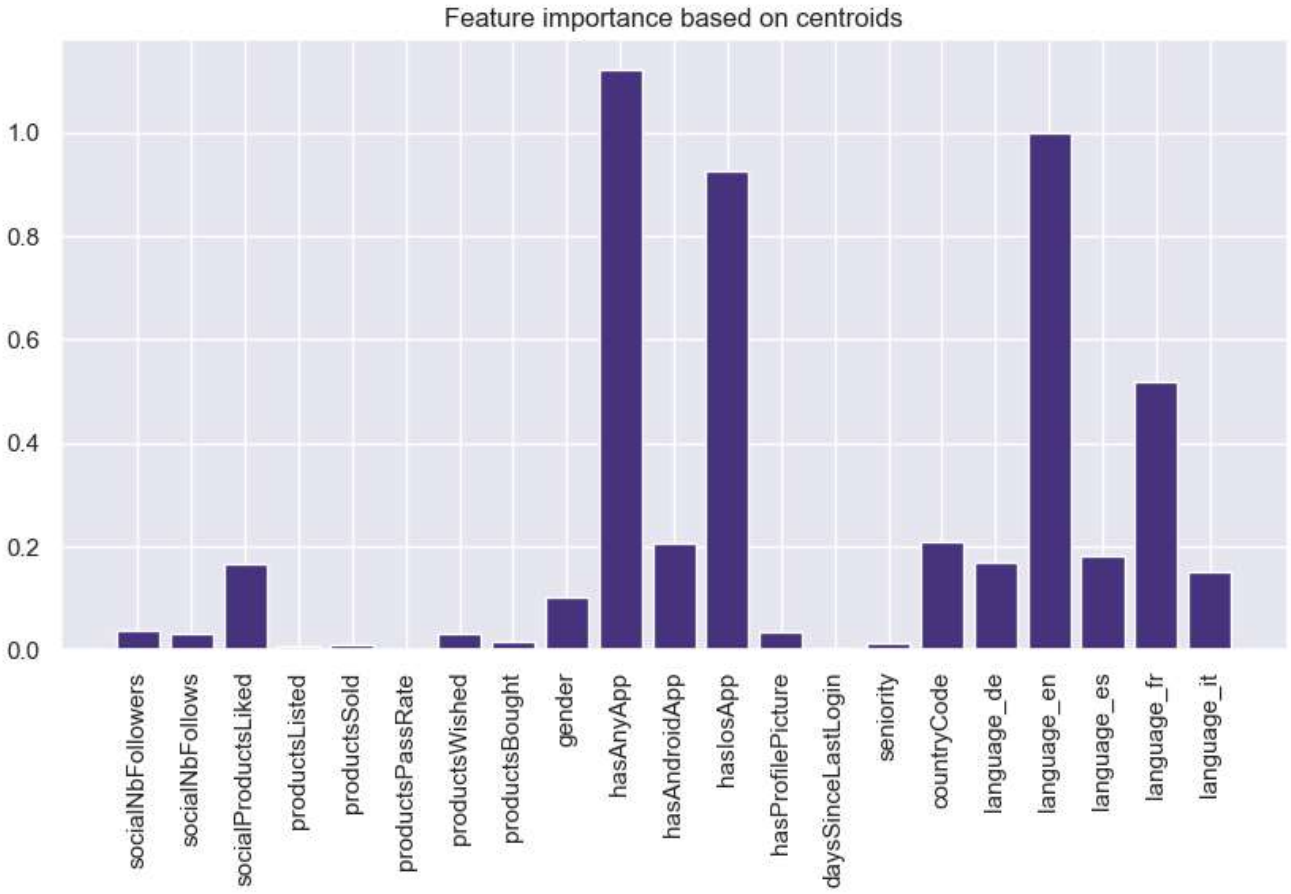


3D PCA Cluster Plot

Model	Silhouette Score	Calinski-Harabasz Score
Agglomerative (K=3)	0.176832	9618.795831

In [444…

```
fi = plot_feature_importance(df, labels)
analyze_features(df, fi, labels, 3)
```



Out[444...

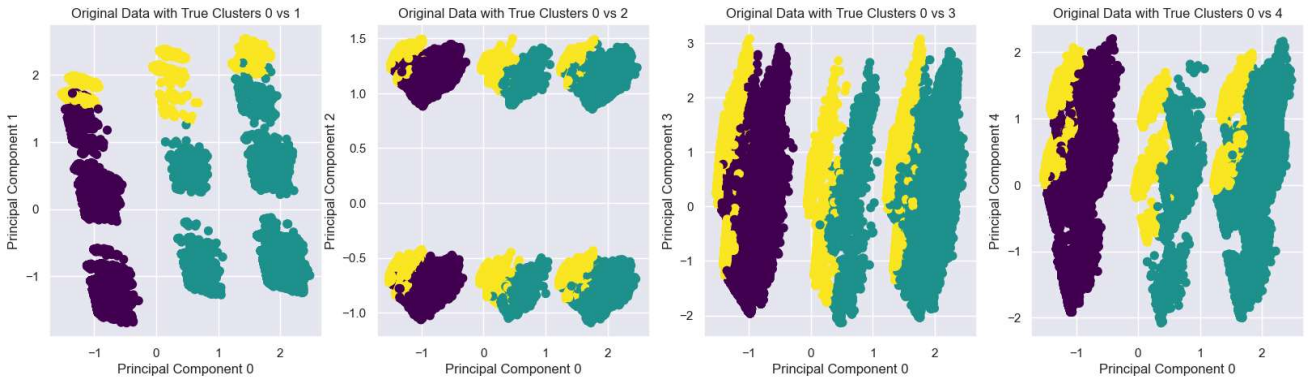
	0	1	2
hasAnyApp	0.993659	-0.896278	-1.0
hasIosApp	0.651131	-0.896278	-1.0
language_en	0.036673	-1.000000	1.0
language_fr	-0.477595	0.036943	-1.0

Interpretacija rezultata:

- **0**: imaju aplikaciju
- **1 i 2**: nemaju aplikaciju:
  - **1**: ne govore engleski
  - **2**: govore engleski

In [53]:
```
nonimportant_columns = df.columns[feature_importance < 0.25]
df_features = df.drop(columns=nonimportant_columns)
```

In [54]:
```
agg_clustering = AgglomerativeClustering(n_clusters=3)
labels = agg_clustering.fit_predict(df_features)

plot_2d(X_pca, labels)
plot_3d(X_pca, labels)
register_results(
    results, 'Agglomerative (K=3, important cols)', df_features, labels)
```

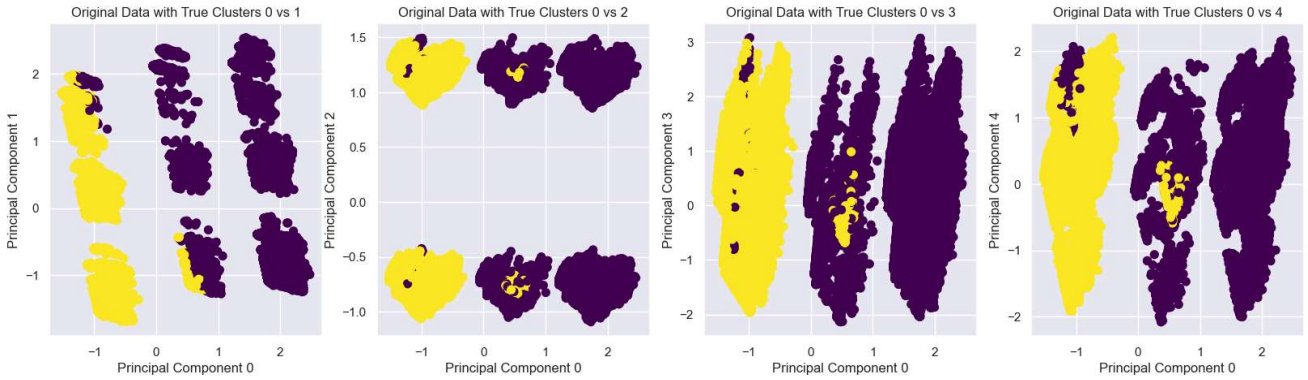


3D PCA Cluster Plot

Model	Silhouette Score	Calinski-Harabasz Score
Agglomerative (K=3, important cols)	0.449925	27185.108948

Fuzzy K-Means

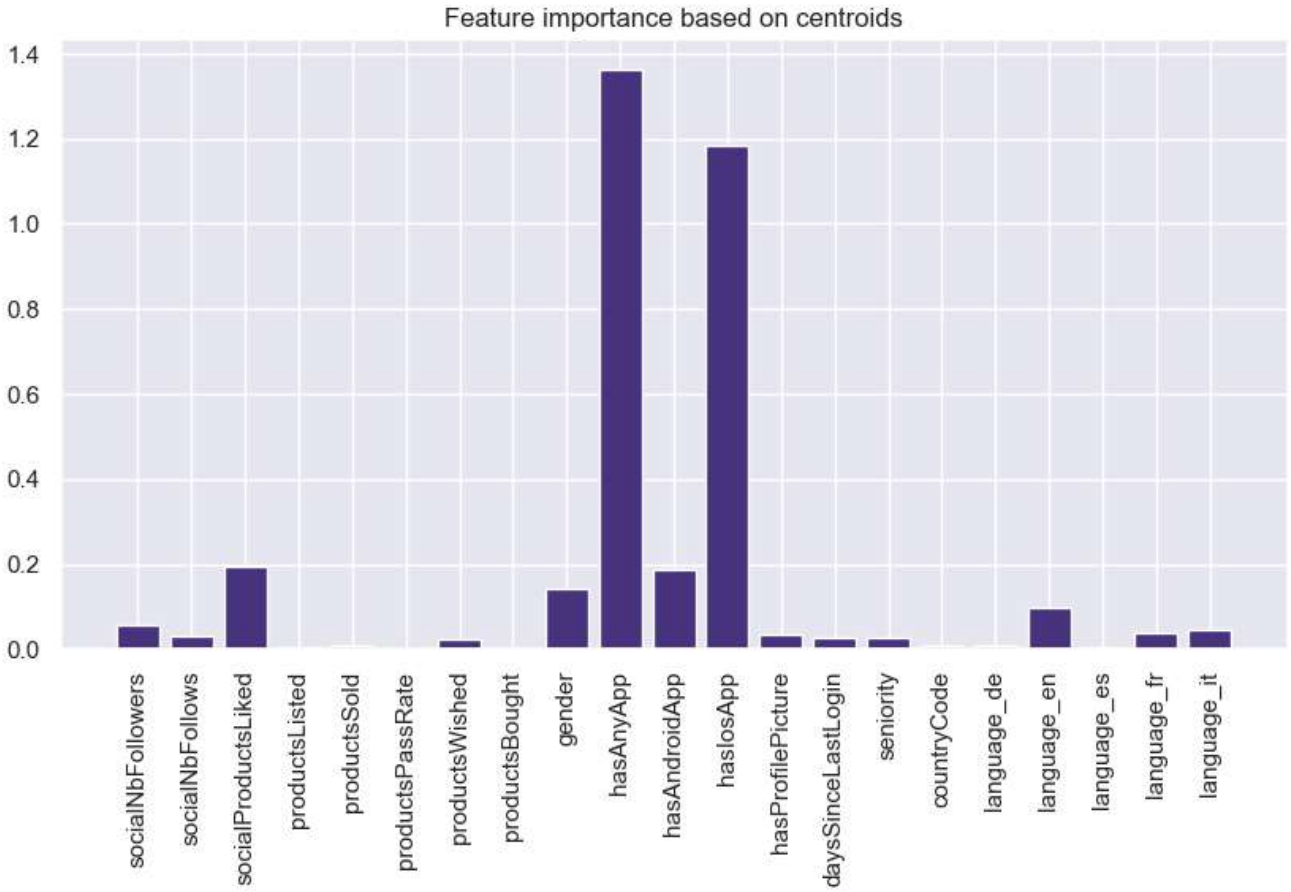
In [55]:
```
import skfuzzy as fuzz

cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(
    df.T, 2, 2, error=0.0005, maxiter=10000, init=None)

cluster_membership = np.argmax(u, axis=0)

plot_2d(X_pca, cluster_membership)
plot_3d(X_pca, cluster_membership)
register_results(results, 'Fuzzy CMeans(K=2)', df, cluster_membership)
```



3D PCA Cluster Plot

Model	Silhouette Score	Calinski-Harabasz Score
Fuzzy CMeans(K=2)	0.204777	10897.056752

In [56]:
```
fi = plot_feature_importance(df, cluster_membership)
analyze_features(df, fi, cluster_membership, 2)
```



Out[56]:

	0	1
hasAnyApp	0.963733	-0.965278
hasIosApp	0.674732	-1.000000

Spectral

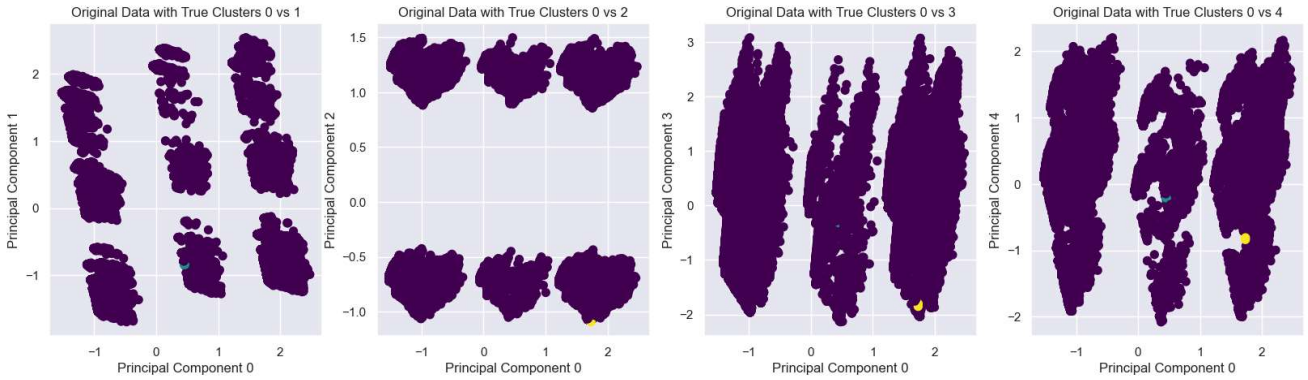
In [57]:
```
spectral = cluster.SpectralClustering(
    n_clusters=3, eigen_solver='arpack', affinity="nearest_neighbors")

spectral.fit(df)
```

Out[57]:
```
SpectralClustering(affinity='nearest_neighbors', eigen_solver='arpack',
                   n_clusters=3)
```

In [58]:
```
y_spectral = spectral.labels_
np.unique(y_spectral)
```

Out[58]:
```
array([0, 1, 2])
```

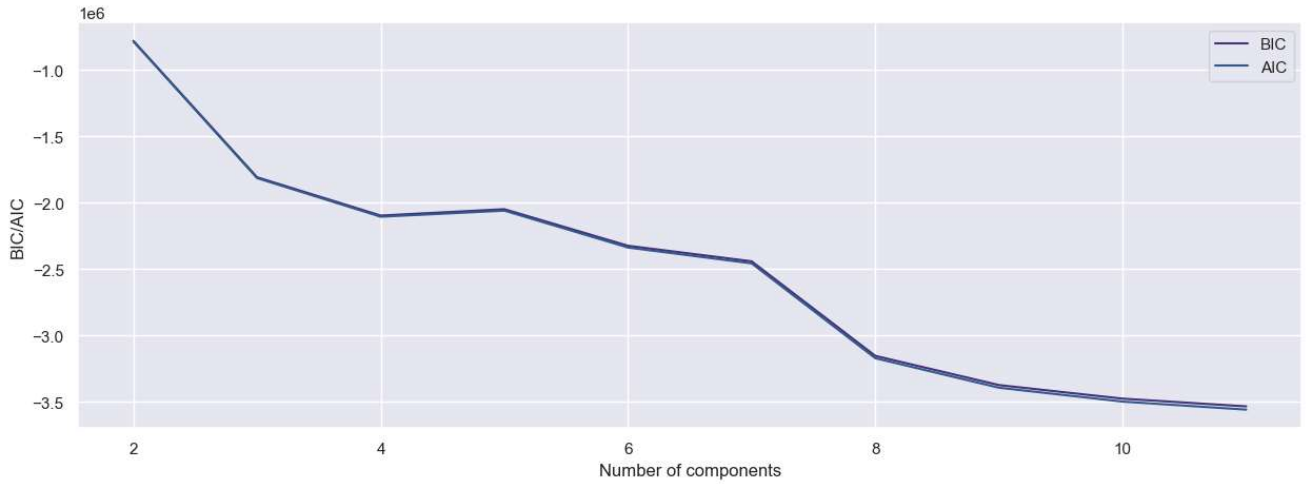
In [61]:
```
plot_2d(X_pca, y_spectral)
register_results(results, 'Spectral (K=3)', df, y_spectral)
```



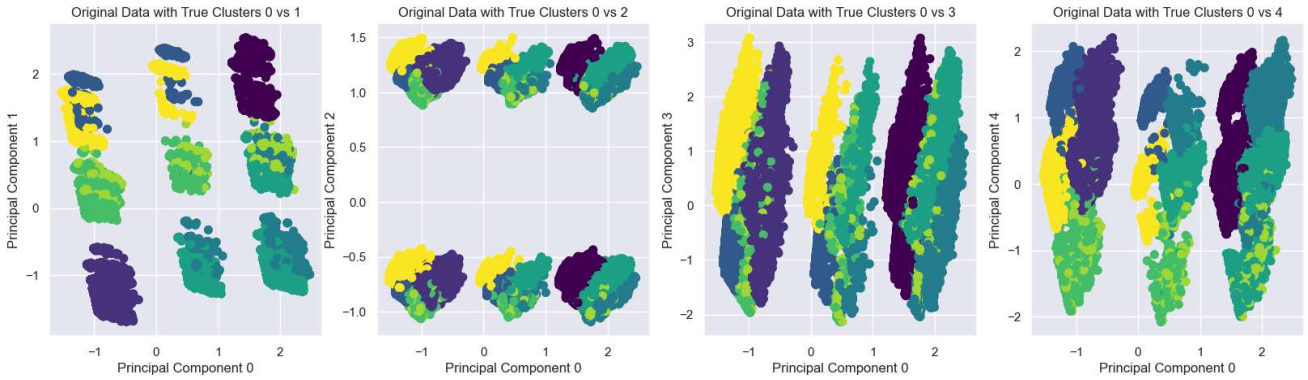
Model	Silhouette Score	Calinski-Harabasz Score
Spectral (K=3)	-0.061996	16.677706

Gaussian Mixture

In [62]:
```
n_components_range = range(2, 12)
models = [GaussianMixture(n, covariance_type='full', random_state=42).fit(df)
          for n in n_components_range]

# plot bic and aic
plt.figure(figsize=(15, 5))
plt.plot(n_components_range, [m.bic(df) for m in models], label='BIC')
plt.plot(n_components_range, [m.aic(df) for m in models], label='AIC')
plt.xlabel('Number of components')
plt.ylabel('BIC/AIC')
plt.legend()
plt.show()
```



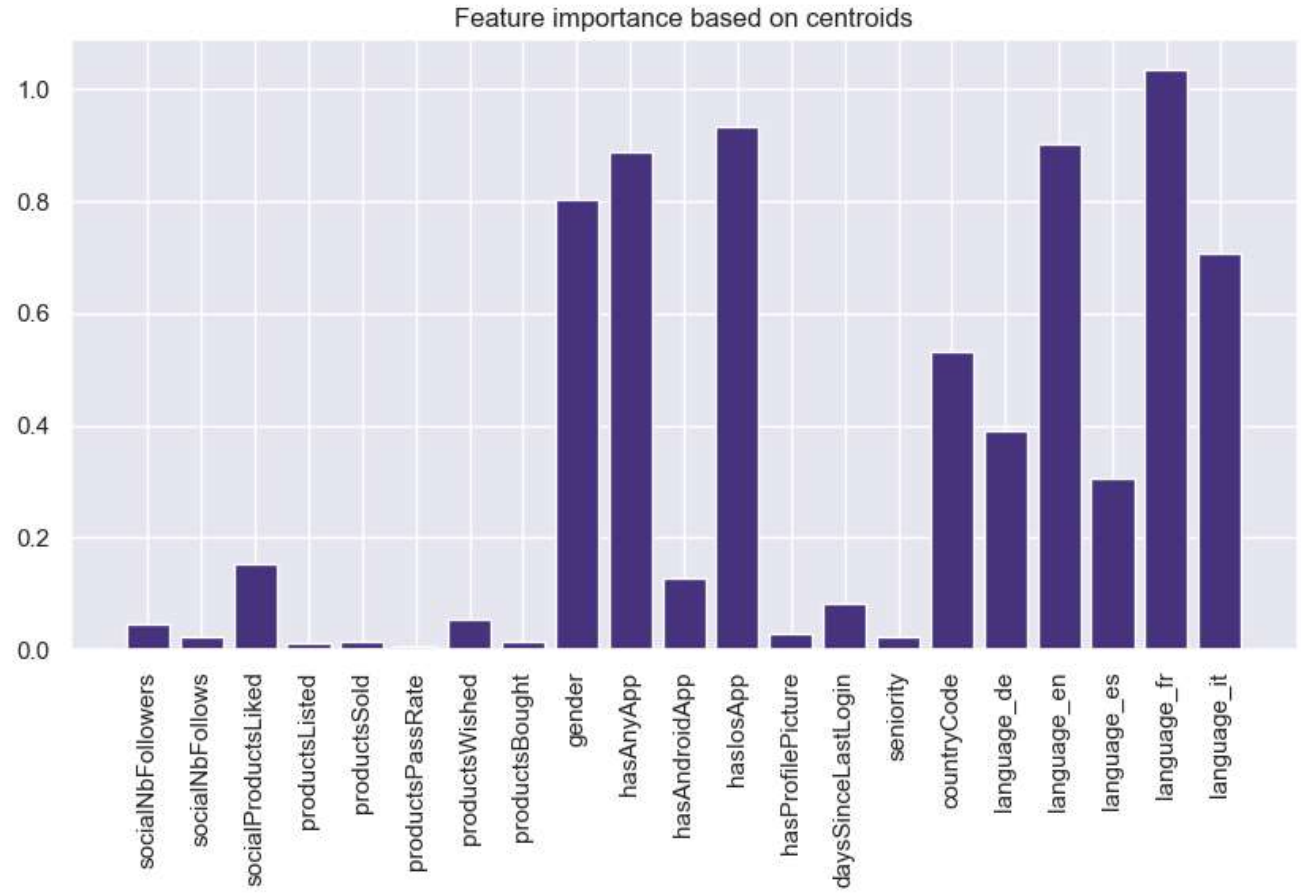
In [82]:
```
from sklearn.mixture import GaussianMixture

gmm = GaussianMixture(n_components=8)
gmm_labels = gmm.fit_predict(df)

plot_2d(X_pca, gmm_labels)
plot_3d(X_pca, gmm_labels)
register_results(results, 'Gaussian Mixture (K=8)', df, gmm_labels)
```



3D PCA Cluster Plot

Model	Silhouette Score	Calinski-Harabasz Score
Gaussian Mixture (K=8)	0.236881	6172.480164

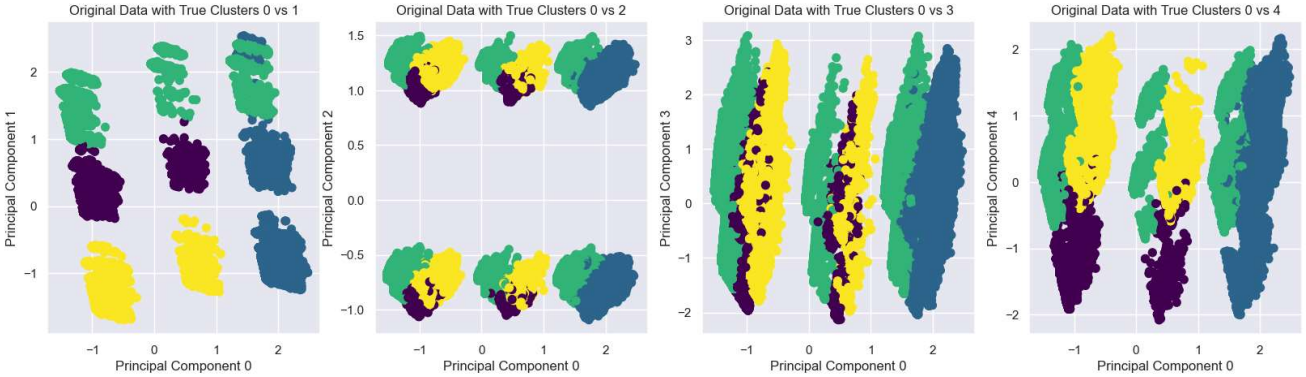
In [84]:
```
fi_k8 = plot_feature_importance(df, gmm_labels)
analyze_features(df, fi_k8, gmm_labels, 8)
```

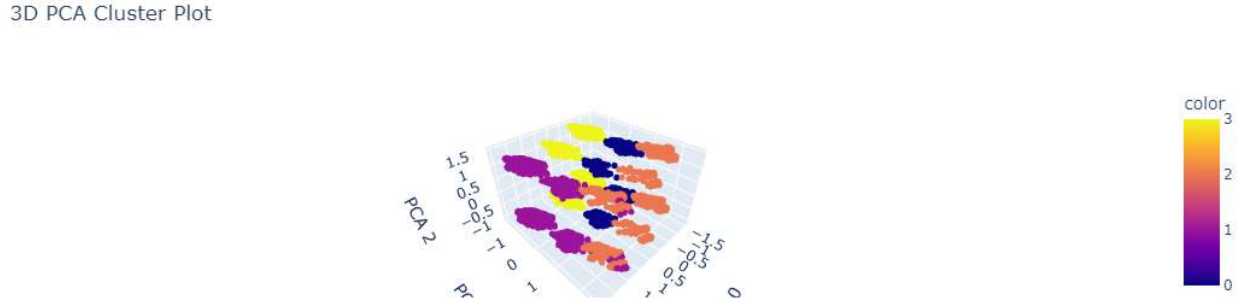


Out[84]:

	0	1	2	3	4	5	6	7
gender	0.339006	0.610913	-1.000000	-1.000000	1.000000	0.516469	0.450507	1.000000
hasAnyApp	1.000000	-1.000000	-0.613438	1.000000	1.000000	-0.734338	-0.006277	-0.772185
hasIosApp	1.000000	-1.000000	-1.000000	0.829358	0.818411	-1.000000	-0.228392	-1.000000
countryCode	0.734398	-0.245287	0.655057	-0.370682	-0.401949	-0.415958	-0.177136	0.683122
language_de	-1.000000	-1.000000	-1.000000	-0.625076	-0.796226	0.118192	-1.000000	-1.000000
language_en	-1.000000	1.000000	-1.000000	0.509480	0.663122	-1.000000	-1.000000	-1.000000
language_es	-1.000000	-1.000000	-1.000000	-0.884404	-0.866895	-0.118192	-1.000000	-1.000000
language_fr	1.000000	-1.000000	1.000000	-1.000000	-1.000000	-1.000000	-1.000000	1.000000
language_it	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	1.000000	-1.000000

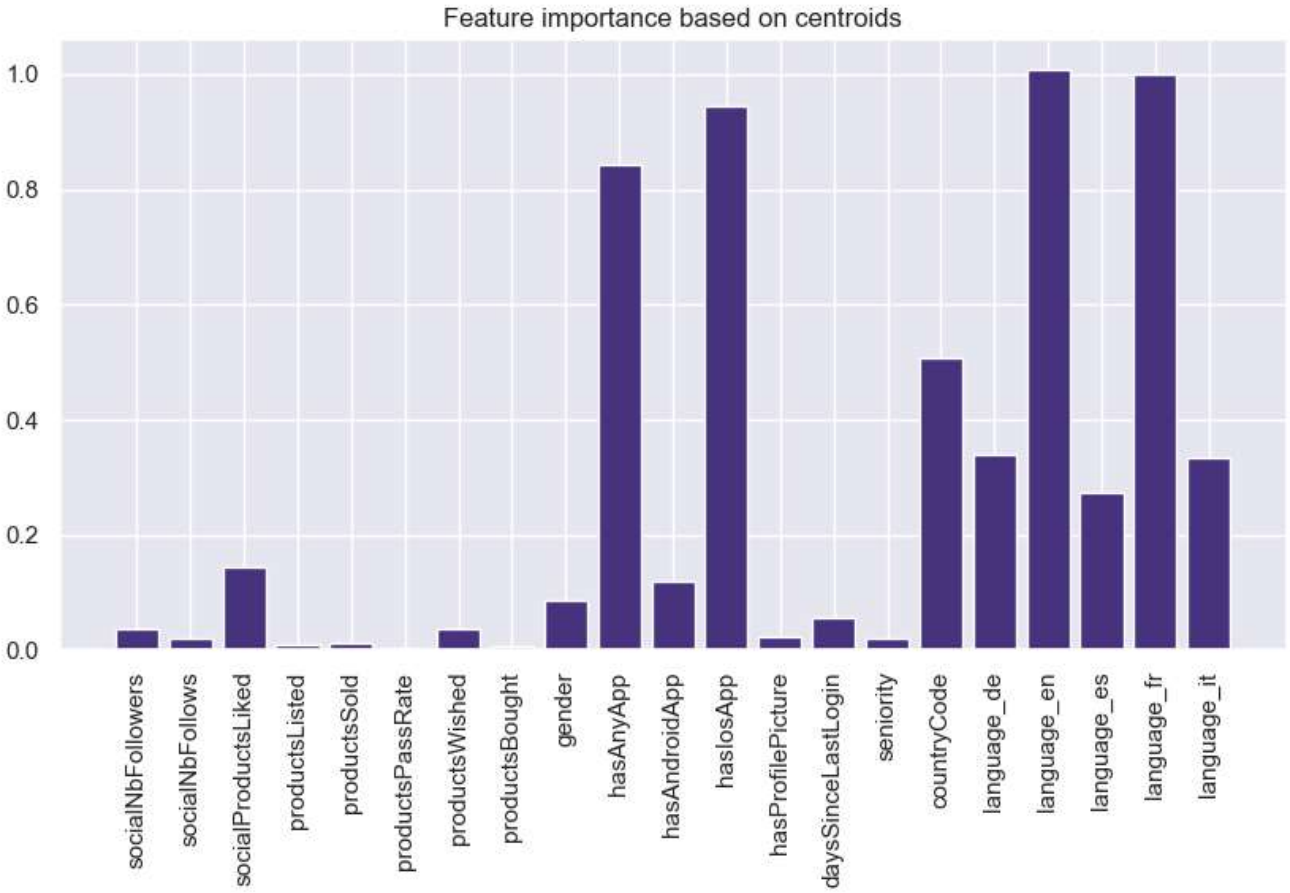
In [79]:
```
gmm = GaussianMixture(n_components=4)
gmm_labels = gmm.fit_predict(df)

plot_2d(X_pca, gmm_labels)
plot_3d(X_pca, gmm_labels)
register_results(results, 'Gaussian Mixture (K=4)', df, gmm_labels)
```





Model	Silhouette Score	Calinski-Harabasz Score
Gaussian Mixture (K=4)	0.218829	9008.029901

In [80]:
```
fi = plot_feature_importance(df, gmm_labels)
analyze_features(df, fi, gmm_labels, 4)
```



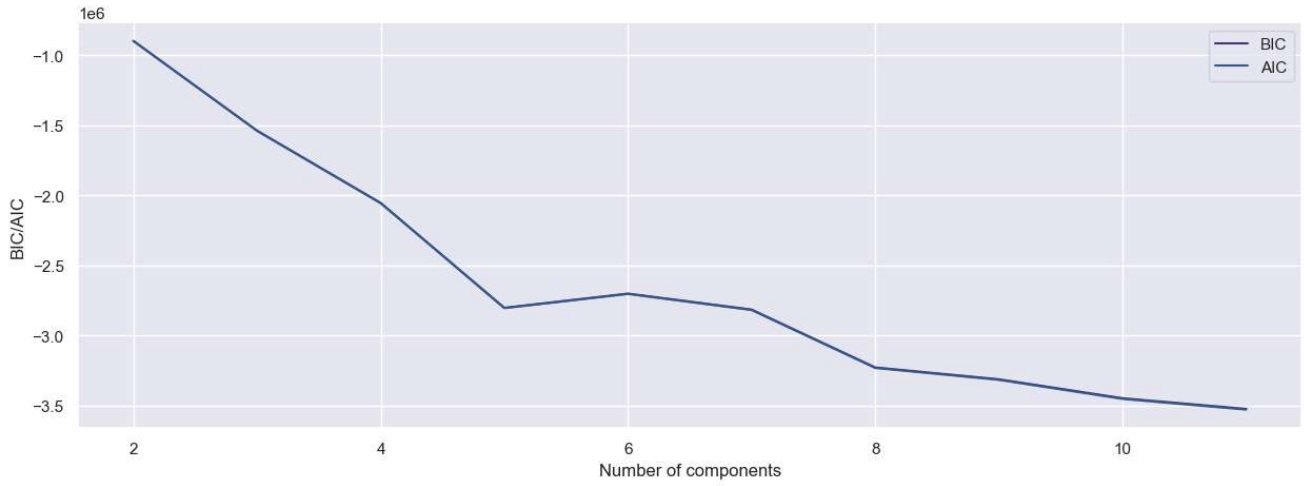
Out[80]:

	0	1	2	3
hasAnyApp	-0.694013	1.000000	-0.201109	-0.871023
hasIosApp	-1.000000	1.000000	-0.390723	-1.000000
countryCode	-0.331281	-0.378427	0.692870	-0.243397
language_de	-0.279379	-0.761063	-1.000000	-1.000000
language_en	-1.000000	0.383954	-1.000000	1.000000
language_es	-0.425721	-0.881089	-1.000000	-1.000000
language_fr	-1.000000	-0.996180	1.000000	-1.000000
language_it	-0.294900	-0.745622	-1.000000	-1.000000

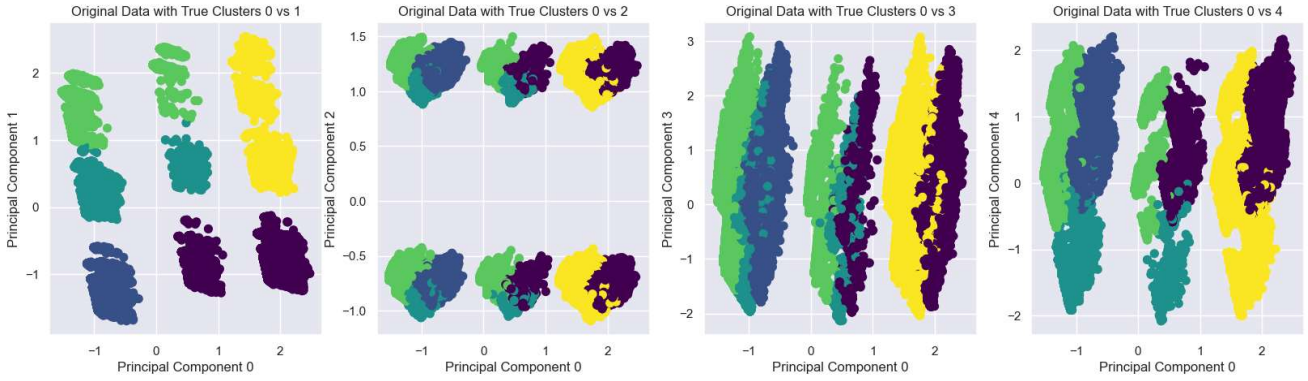
In [85]:
```
nonimportant_columns = df.columns[fi_k8 < 0.2]
df_features = df.drop(columns=nonimportant_columns)
```

In [86]:
```
n_components_range = range(2, 12)
models = [GaussianMixture(n, covariance_type='full', random_state=42).fit(df_features)
          for n in n_components_range]

# plot bic and aic
plt.figure(figsize=(15, 5))
plt.plot(n_components_range, [m.bic(df_features) for m in models], label='BIC')
plt.plot(n_components_range, [m.aic(df_features) for m in models], label='AIC')
plt.xlabel('Number of components')
```

```
plt.ylabel('BIC/AIC')
plt.legend()
plt.show()
```



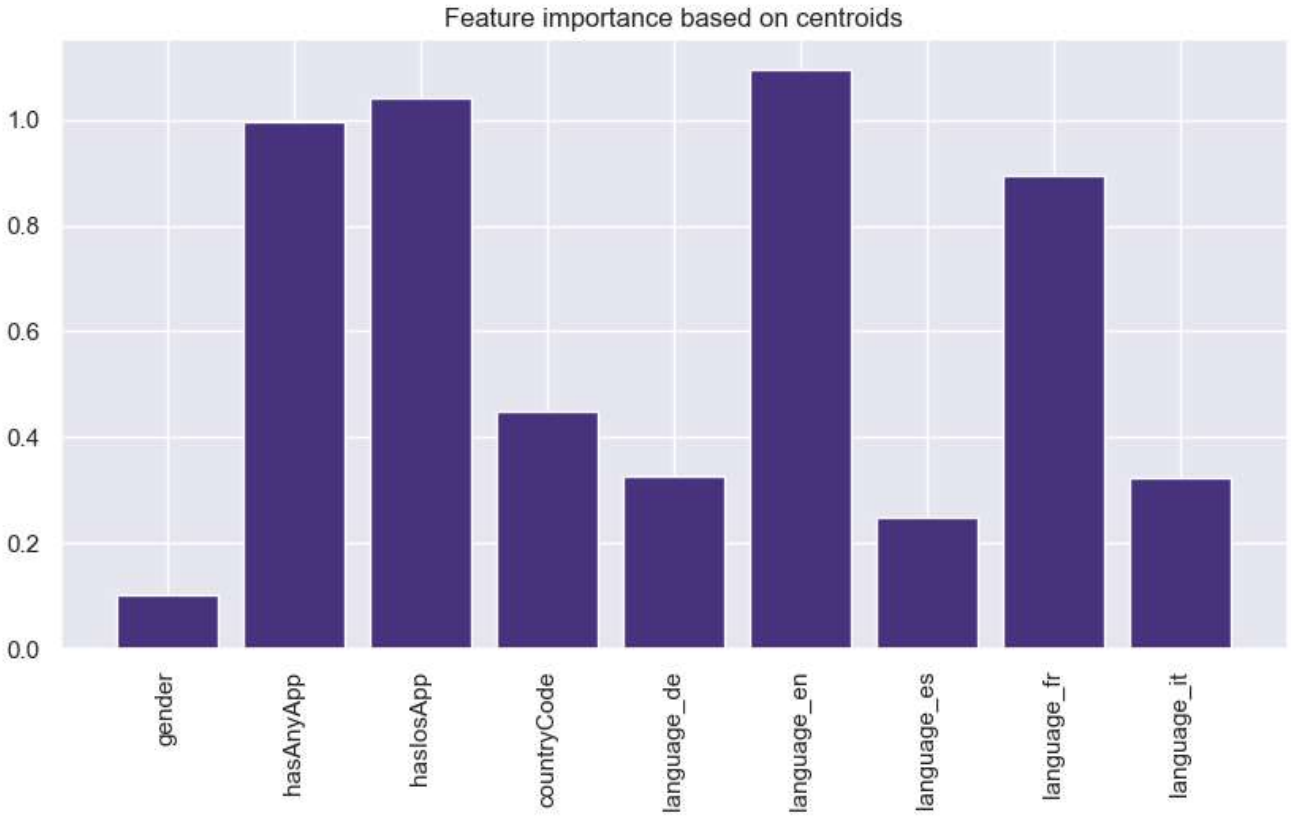
In [92]:
```
gmm = GaussianMixture(n_components=5)
gmm_labels = gmm.fit_predict(df_features)

plot_2d(X_pca, gmm_labels)
plot_3d(X_pca, gmm_labels)
register_results(results, 'Gaussian Mixture (K=5, important cols)',
                 df_features, gmm_labels)
```



3D PCA Cluster Plot

Model	Silhouette Score	Calinski-Harabasz Score
Gaussian Mixture (K=5, important cols)	0.463167	20108.117325

In [93]:
```
fi = plot_feature_importance(df_features, gmm_labels)
analyze_features(df_features, fi, gmm_labels, 5)
```



Out[93]:

	0	1	2	3	4
hasAnyApp	1.000000	-1.000000	-0.694013	-0.727315	1.000000
hasIosApp	0.785216	-1.000000	-1.000000	-1.000000	1.000000
countryCode	-0.382319	-0.245287	-0.331281	0.675190	0.195004
language_de	-1.000000	-1.000000	-0.279379	-1.000000	-0.605363
language_en	1.000000	1.000000	-1.000000	-1.000000	-1.000000
language_fr	-1.000000	-1.000000	-1.000000	1.000000	-0.011174
language_it	-1.000000	-1.000000	-0.294900	-1.000000	-0.579861

DBSCAN

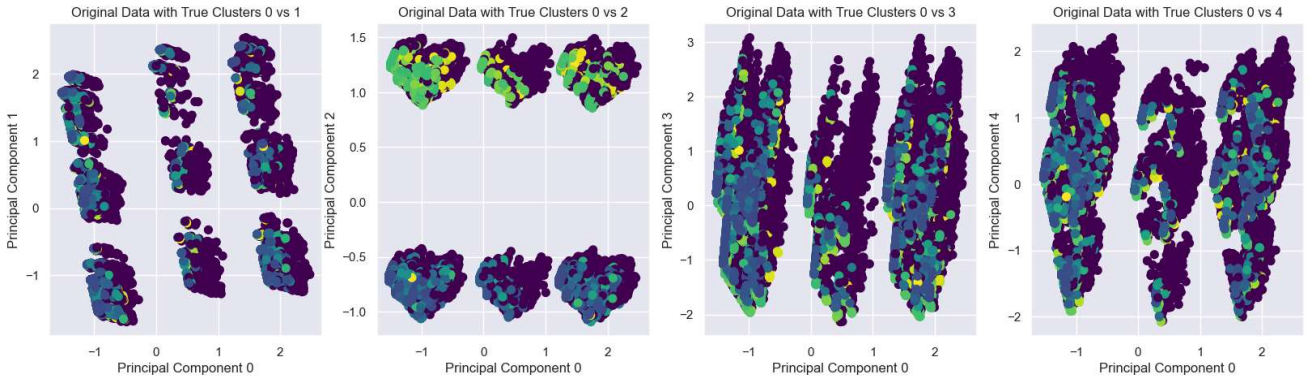
In [94]:
```
from sklearn.cluster import DBSCAN
from sklearn.model_selection import GridSearchCV
param_grid = {
    'eps': [0.1, 0.2, 0.5, 0.7, 1, 1.5, 2, 2.5, 3],
    'min_samples': [3, 5, 15, 30]
}
dbscan = DBSCAN()
dbscan_cv = GridSearchCV(dbscan, param_grid, scoring='adjusted_rand_score')
dbscan_cv.fit(df)
dbscan_cv.best_params_
```

Out[94]:
```
{'eps': 0.1, 'min_samples': 3}
```

In [95]:
```
dbscan = DBSCAN(eps=0.1, min_samples=3)
dbscan_labels = dbscan.fit_predict(df)
```

```
print(f"Number of clusters: {len(np.unique(dbscan_labels))}")
plot_2d(X_pca, dbscan_labels)
```

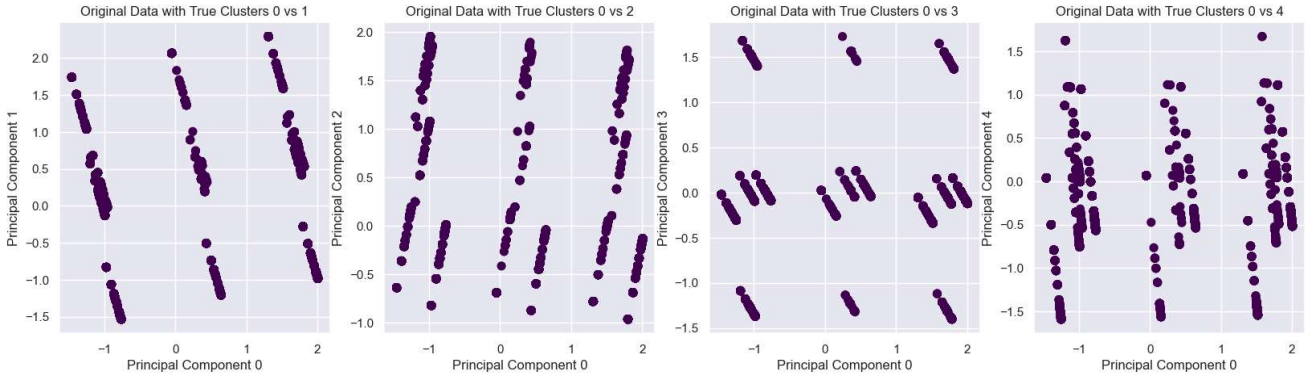
```
Number of clusters: 1621
```



Mean Shift

In [486…

```
from sklearn.cluster import MeanShift

ms = MeanShift(max_iter=2)
ms_labels = ms.fit_predict(df)
plot_2d(X_pca, ms_labels)
```



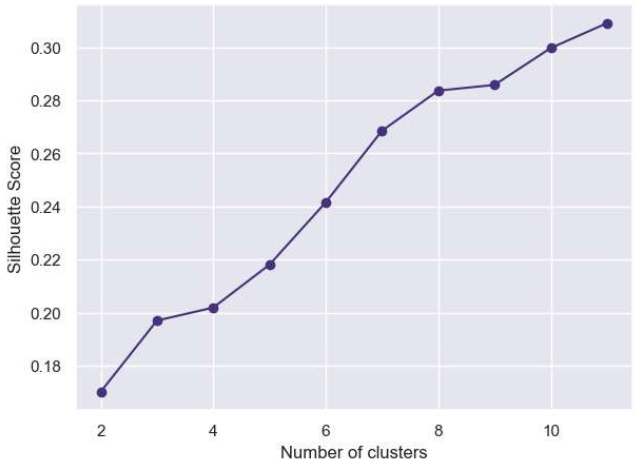
Birch

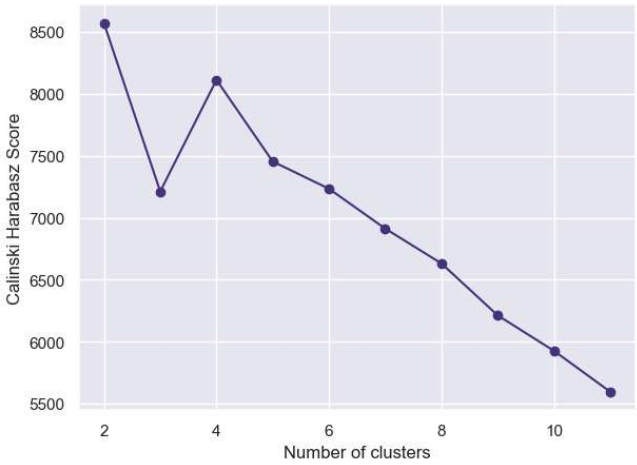
In [96]:

```
from sklearn.cluster import Birch


silhouette = []
calinski = []
for k in range(2, 12):
    birch = Birch(n_clusters=k)
    birch_labels = birch.fit_predict(df)
    silhouette.append(metrics.silhouette_score(
        df, birch_labels, sample_size=5000))
    calinski.append(metrics.calinski_harabasz_score(df, birch_labels))
    print(k, end=' ')

plt.figure(figsize=(15, 5))
plt.subplot(1, 2, 1)
plt.plot(range(2, 12), silhouette, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')
plt.subplot(1, 2, 2)
plt.plot(range(2, 12), calinski, marker='o')
```

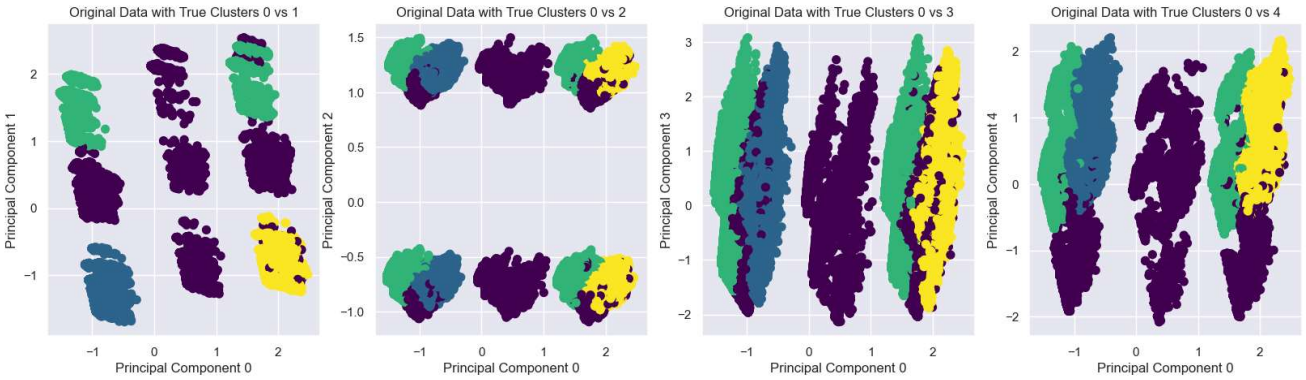
```
plt.xlabel('Number of clusters')
plt.ylabel('Calinski Harabasz Score')
plt.show()
```

```
2 3 4 5 6 7 8 9 10 11
```





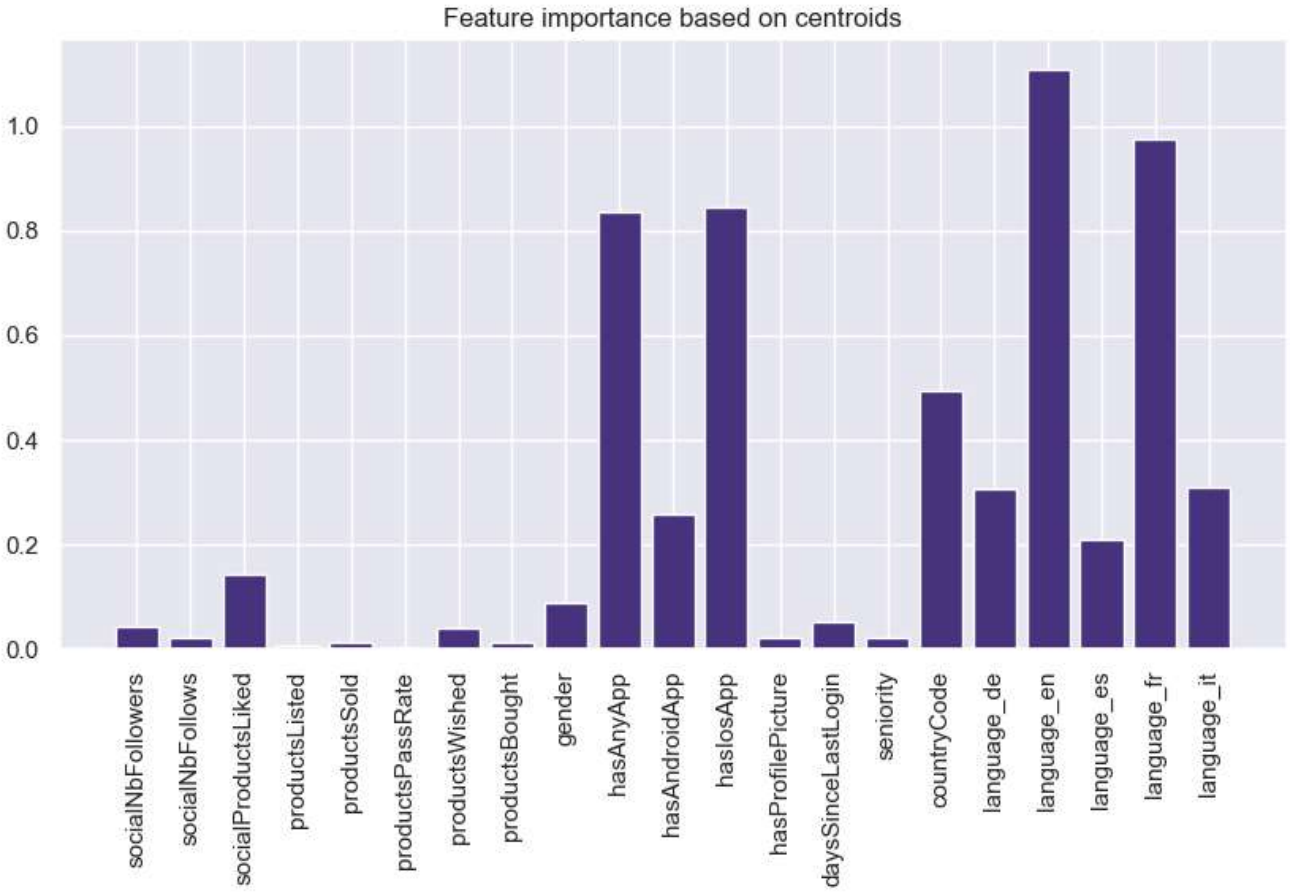
In [98]:
```
birch = Birch(n_clusters=4)
birch_labels = birch.fit_predict(df)
plot_2d(X_pca, birch_labels)
plot_3d(X_pca, birch_labels)
register_results(results, 'Birch (K=4)', df, birch_labels)
```



3D PCA Cluster Plot

Model	Silhouette Score	Calinski-Harabasz Score
Birch (K=4)	0.19966	8112.911935

In [99]:
```
fi = plot_feature_importance(df, birch_labels)
analyze_features(df, fi, birch_labels, 5)
```



Out[99]:

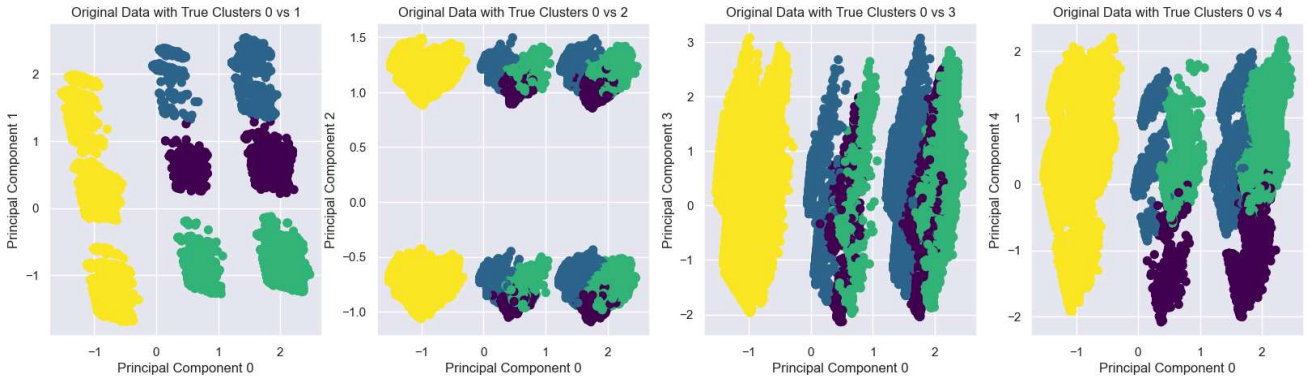
	0	1	2	3	4
hasAnyApp	0.085098	-1.000000	-0.326790	1.000000	NaN
hasAndroidApp	-0.486266	-1.000000	-0.999820	-0.999536	NaN
hasIosApp	-0.410823	-1.000000	-0.326790	1.000000	NaN
countryCode	-0.223374	-0.245287	0.683582	-0.403181	NaN
language_de	-0.386124	-1.000000	-1.000000	-1.000000	NaN
language_en	-0.833396	1.000000	-1.000000	1.000000	NaN
language_fr	-0.822468	-1.000000	1.000000	-1.000000	NaN
language_it	-0.379987	-1.000000	-1.000000	-1.000000	NaN

In [100…

```
nonimportant_columns = df.columns[fi < 0.2]
df_features = df.drop(columns=nonimportant_columns)
```

In [101…

```
birch = Birch(n_clusters=4)
birch_labels = birch.fit_predict(df_features)
plot_2d(X_pca, birch_labels)
plot_3d(X_pca, birch_labels)
register_results(results, 'Birch (K=4, important cols)',
                 df, birch_labels)
```

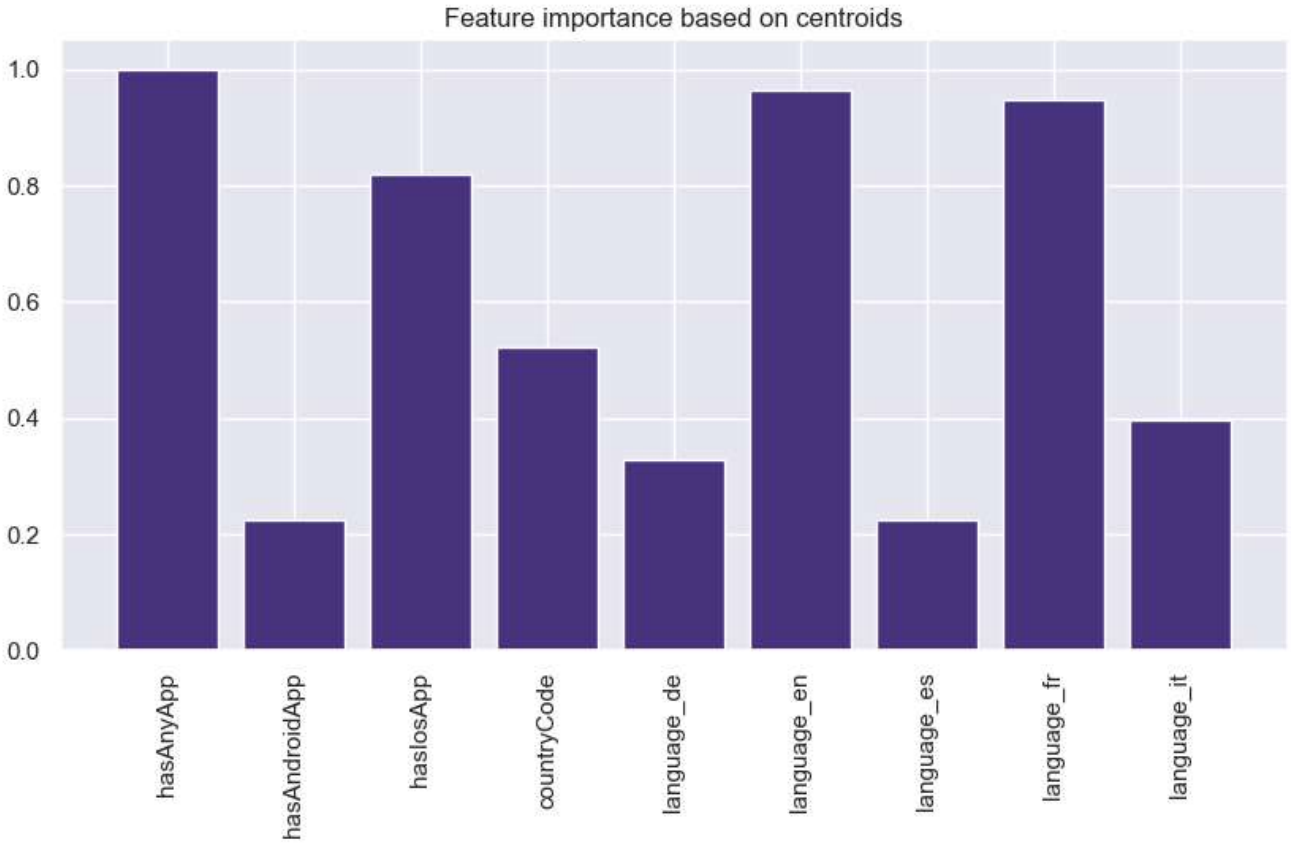


3D PCA Cluster Plot

Model	Silhouette Score	Calinski-Harabasz Score
Birch (K=4, important cols)	0.191999	6783.58443

In [102…

```
fi = plot_feature_importance(df_features, birch_labels)
analyze_features(df_features, fi, birch_labels, 5)
```



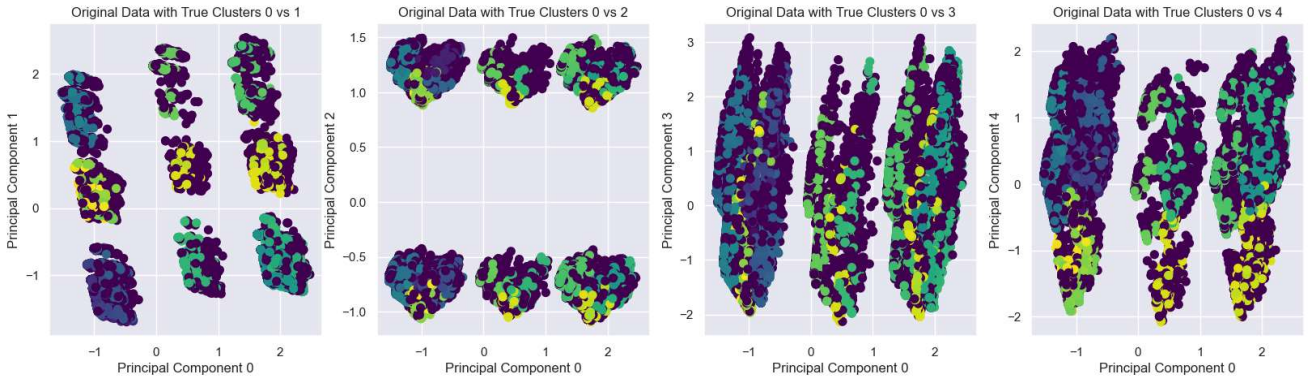
Out[102…

	0	1	2	3	4
hasAnyApp	1.000000	1.000000	1.000000	-1.000000	NaN
hasIosApp	0.553939	0.527620	0.785216	-1.000000	NaN
countryCode	-0.334991	0.745193	-0.382319	-0.030743	NaN
language_de	-0.307475	-1.000000	-1.000000	-0.833386	NaN
language_en	-1.000000	-1.000000	1.000000	0.059156	NaN
language_fr	-1.000000	1.000000	-1.000000	-0.485778	NaN
language_it	-0.168485	-1.000000	-1.000000	-0.854535	NaN

OPTICS

In [103…

```
from sklearn.cluster import OPTICS

optics = OPTICS()
optics_labels = optics.fit_predict(df)

plot_2d(X_pca, optics_labels)
```



In [104…

```
np.unique(optics_labels)
```

Out[104…

```
array([  -1,    0,    1, ..., 3179, 3180, 3181])
```

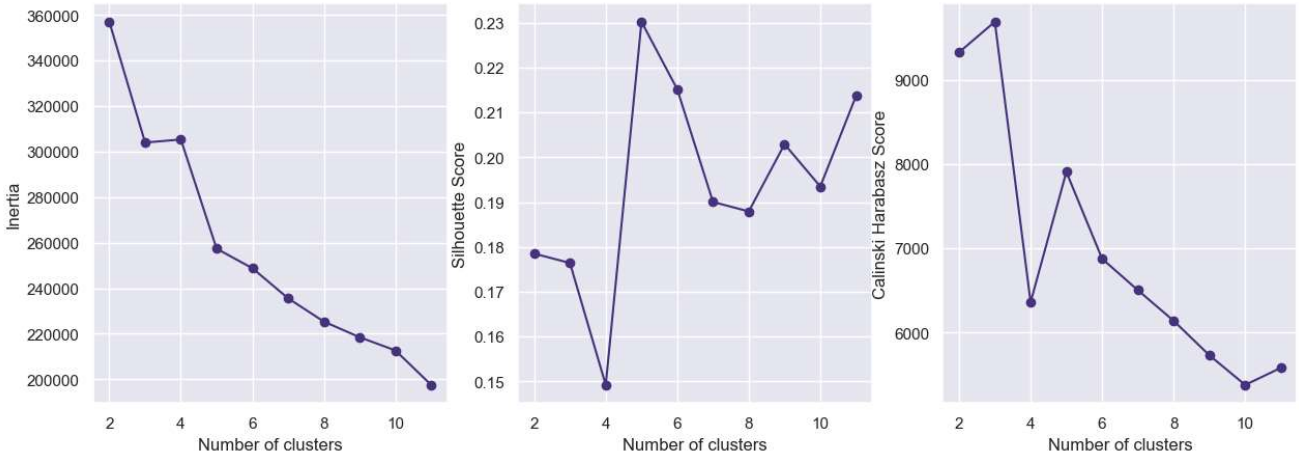
Bisecting K-Means

In [105…

```
from sklearn.cluster import BisectingKMeans
inertia = []
silhouette = []
calinski = []
for k in range(2, 12):
    bkm = BisectingKMeans(n_clusters=k)
    bkm_labels = bkm.fit_predict(df)
    inertia.append(bkm.inertia_)
    silhouette.append(metrics.silhouette_score(
        df, bkm_labels, sample_size=5000))
    calinski.append(metrics.calinski_harabasz_score(df, bkm_labels))
    print(k, end=' ')

plt.figure(figsize=(15, 5))
```

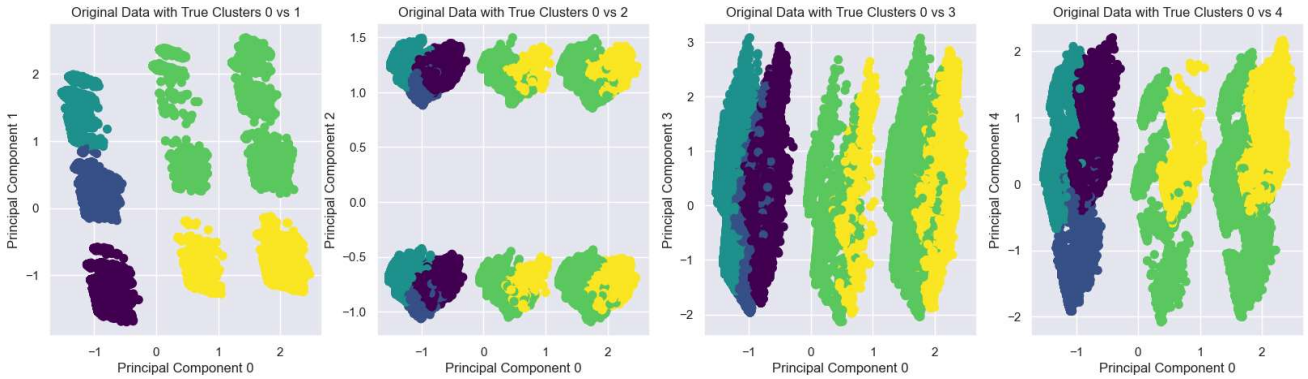
```
plt.subplot(1, 3, 1)
plt.plot(range(2, 12), inertia, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.subplot(1, 3, 2)
plt.plot(range(2, 12), silhouette, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')
plt.subplot(1, 3, 3)
plt.plot(range(2, 12), calinski, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Calinski Harabasz Score')
plt.show()
```

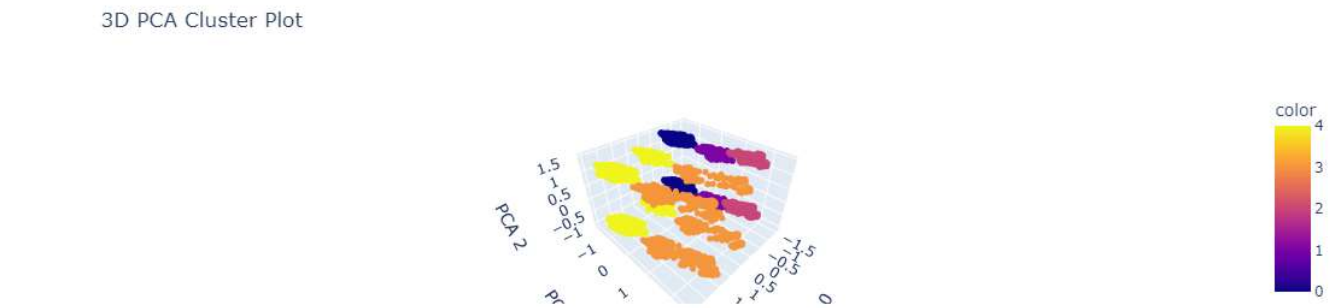
```
2 3 4 5 6 7 8 9 10 11
```



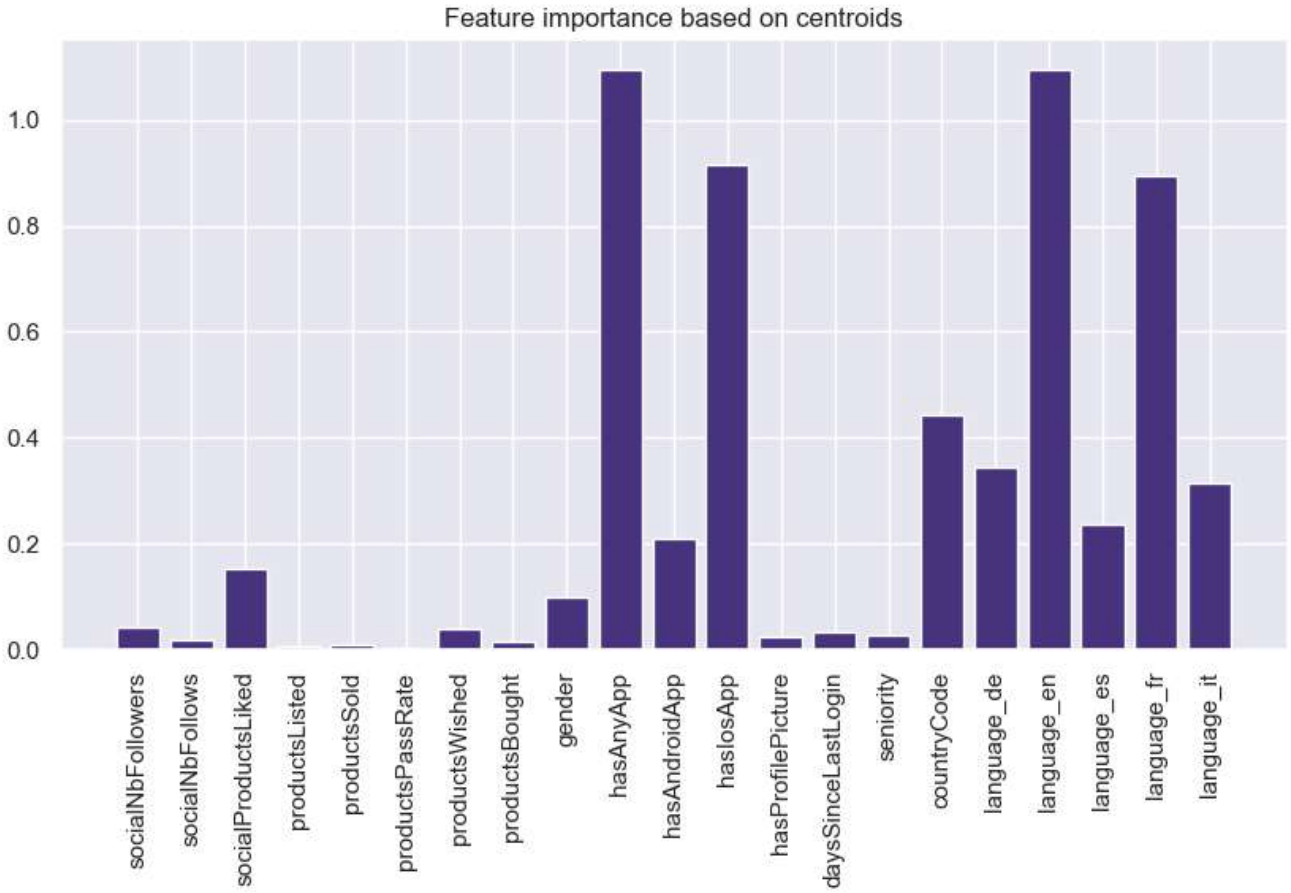
In [107…

```
bkmeans = BisectingKMeans(n_clusters=5)
bkmeans_labels = bkmeans.fit_predict(df)

plot_2d(X_pca, bkmeans_labels)
plot_3d(X_pca, bkmeans_labels)
register_results(results, 'Bisecting KMeans (K=5)', df, bkmeans_labels)

fi = plot_feature_importance(df, bkmeans_labels)
analyze_features(df, fi, bkmeans_labels, 5)
```





Model	Silhouette Score	Calinski-Harabasz Score
Bisecting KMeans (K=5)	0.23132	8174.36548



Out[107…

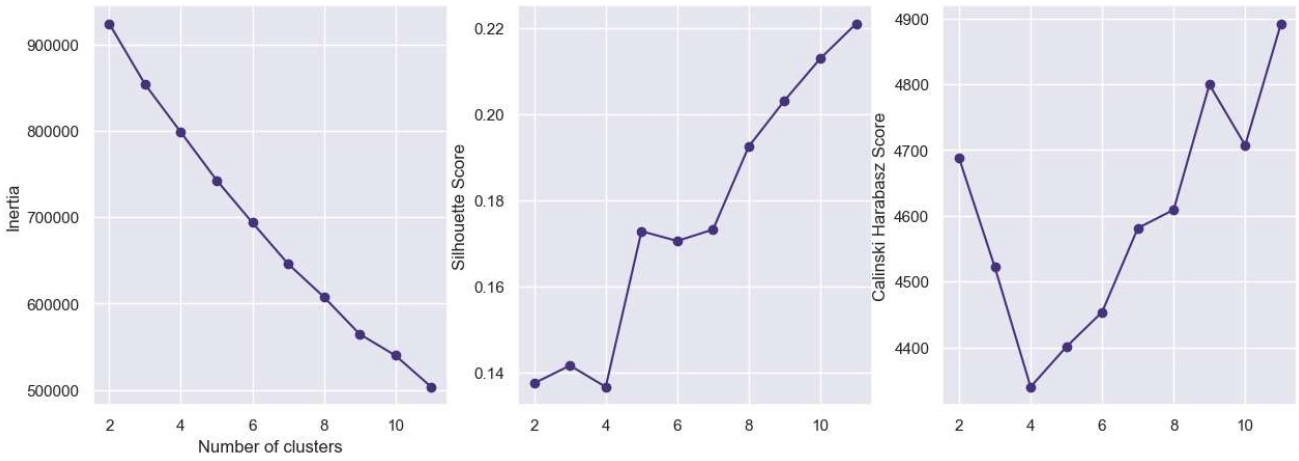
	0	1	2	3	4
hasAnyApp	-1.000000	-1.000000	-1.000000	1.000000	1.000000
hasIosApp	-1.000000	-1.000000	-1.000000	0.540814	0.785216
countryCode	-0.245287	-0.329025	0.658627	0.203679	-0.382319
language_de	-1.000000	-0.218914	-1.000000	-0.652826	-1.000000
language_en	1.000000	-1.000000	-1.000000	-1.000000	1.000000
language_fr	-1.000000	-1.000000	1.000000	-0.002633	-1.000000
language_it	-1.000000	-0.318063	-1.000000	-0.583148	-1.000000

Standardizacija umesto min-max

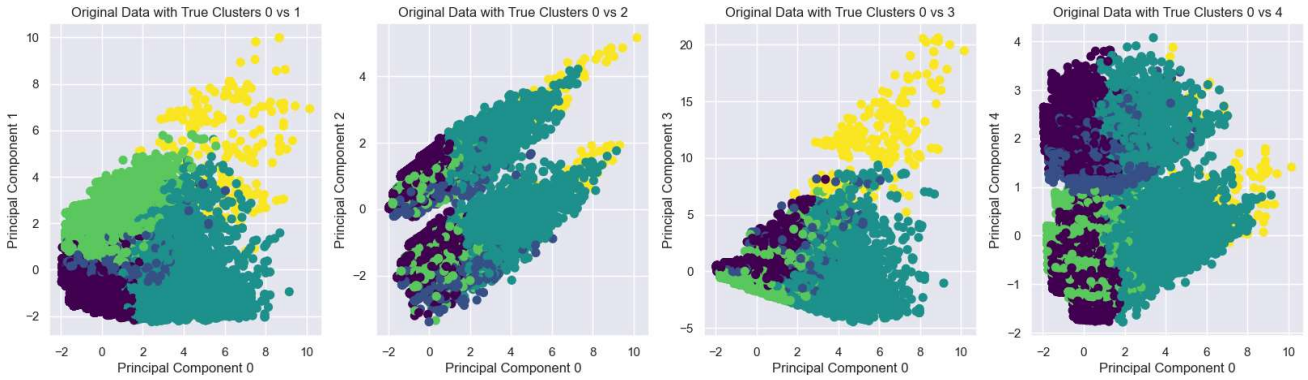
In [108…
```
scaler2 = StandardScaler()
df2 = pd.DataFrame(scaler2.fit_transform(df), columns=df.columns)
```

In [109…
```
k_range = range(2, 12)
inertia = []
silhouette = []
calinski = []
for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df2)
    inertia.append(kmeans.inertia_)
    silhouette.append(metrics.silhouette_score(
        df2, kmeans.labels_, sample_size=10000))
    calinski.append(metrics.calinski_harabasz_score(df2, kmeans.labels_))
    print(k, end=' ')

plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.plot(k_range, inertia, marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.subplot(1, 3, 2)
plt.plot(k_range, silhouette, marker='o')
plt.ylabel('Silhouette Score')
plt.subplot(1, 3, 3)
plt.plot(k_range, calinski, marker='o')
plt.ylabel('Calinski Harabasz Score')
plt.show()
```

```
2 3 4 5 6 7 8 9 10 11
```



In [113…
```
pca = PCA(n_components=5)
X_pca = pca.fit_transform(df2)

NUM_CLUSTERS = 5
kmeans = KMeans(n_clusters=NUM_CLUSTERS, random_state=42)
clusters = kmeans.fit_predict(df2)

plot_2d(X_pca, clusters)
plot_3d(X_pca, clusters)
register_results(results, 'KMeans(K=5, standardization)', df2, clusters)
```

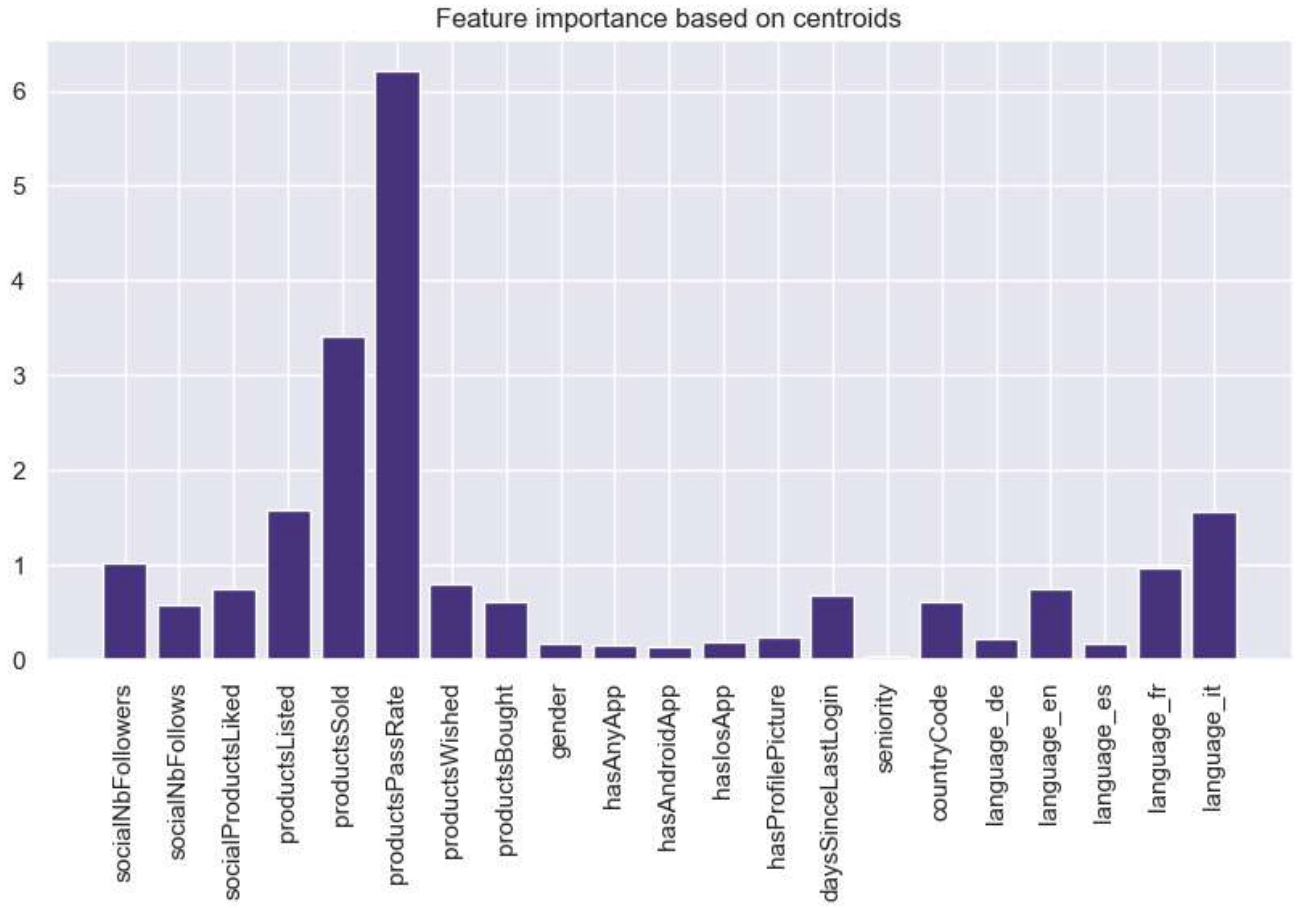


3D PCA Cluster Plot

Model	Silhouette Score	Calinski-Harabasz Score
KMeans(K=5, standardization)	0.169814	4400.753985

In [114…

```
fi = plot_feature_importance(df2, clusters)
analyze_features(df2, fi, clusters, NUM_CLUSTERS, 1)
```



Out[114…

	0	1	2	3	4
socialNbFollowers	-0.187060	-0.106259	1.066091	-0.095708	2.143910
productsListed	-0.099697	0.035749	0.275338	0.013872	3.575418
productsSold	-0.113631	-0.022079	0.209275	0.018065	7.670034
productsPassRate	-0.065686	-0.065686	-0.065686	-0.065686	13.839924
language_it	-0.306382	3.263904	-0.219776	-0.306382	-0.087154

Selekcija najboljeg algoritma

In [116…

```
results.sort_values(by='Silhouette Score', ascending=False)
```

Out[116…

Model	Silhouette Score	Calinski-Harabasz Score
KMeans(K=4,important cols)	0.520517	27729.828964
Gaussian Mixture (K=5, important cols)	0.463167	20108.117325
Agglomerative (K=3, important cols)	0.449925	27185.108948
Hierarchical (dendrogram cut height = 175)	0.248036	7675.246798
Gaussian Mixture (K=8)	0.236881	6172.480164
Bisecting KMeans (K=5)	0.23132	8174.36548
KMeans(K=4)	0.224022	9022.603463
Gaussian Mixture (K=4)	0.218829	9008.029901
Hierarchical (dendrogram cut height = 350)	0.209615	11347.575582
Fuzzy CMeans(K=2)	0.204777	10897.056752
Birch (K=4)	0.19966	8112.911935
Hierarchical (dendrogram cut height = 220)	0.197746	8538.205959
Birch (K=4, important cols)	0.191999	6783.58443
Agglomerative (K=3)	0.176832	9618.795831
Hierarchical (dendrogram cut height = 250)	0.176792	9618.795831
KMeans(K=5, standardization)	0.169814	4400.753985
Spectral (K=3)	-0.061996	16.677706

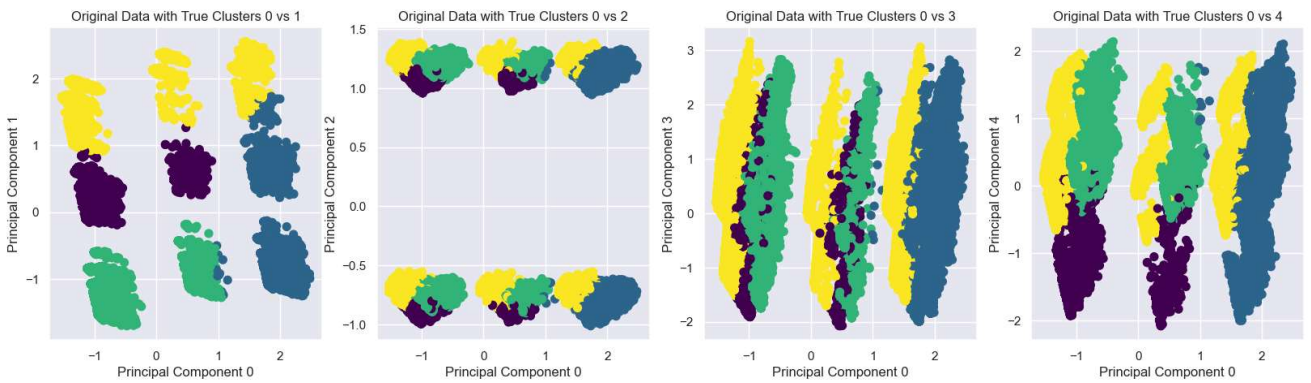
In [117…

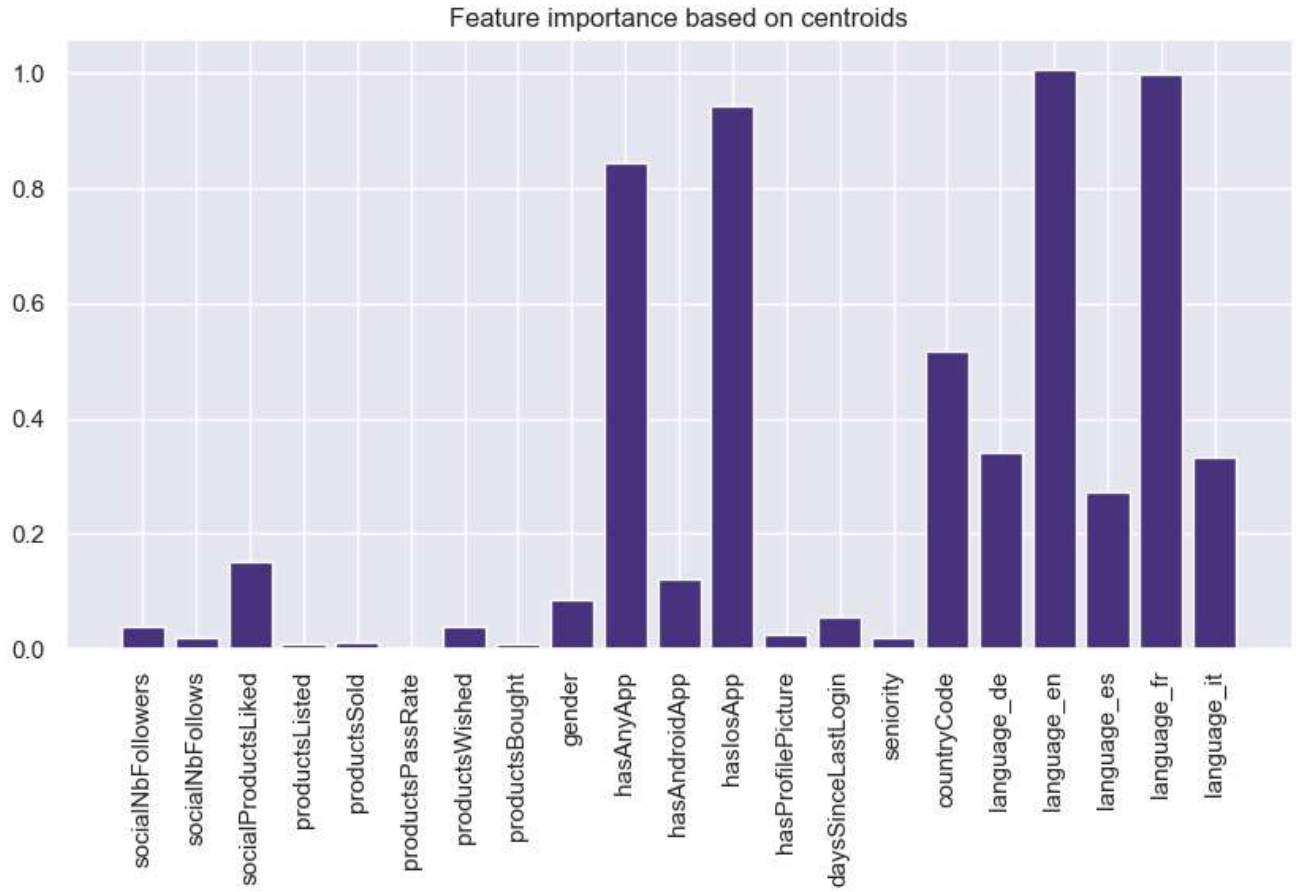
```
results.sort_values(by='Calinski-Harabasz Score', ascending=False)
```

Out[117...

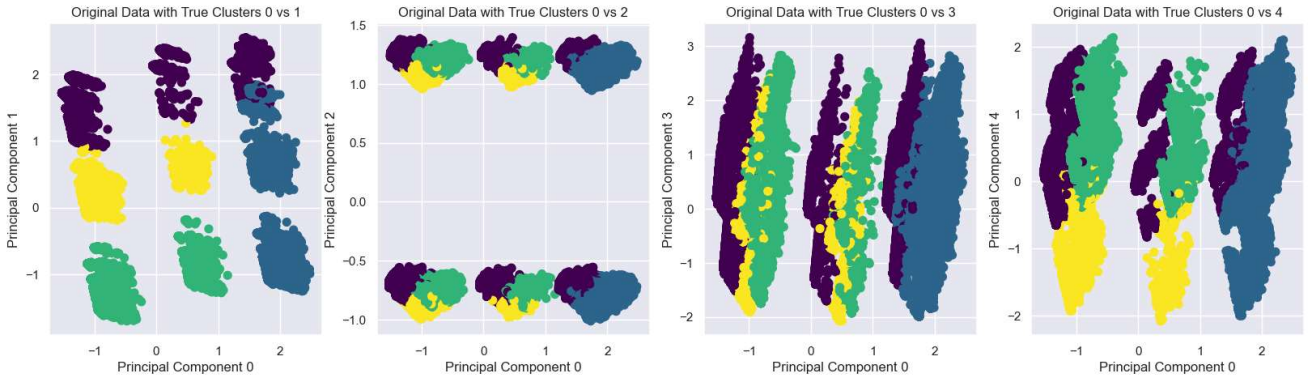
Model	Silhouette Score	Calinski-Harabasz Score
KMeans(K=4,important cols)	0.520517	27729.828964
Agglomerative (K=3, important cols)	0.449925	27185.108948
Gaussian Mixture (K=5, important cols)	0.463167	20108.117325
Hierarchical (dendrogram cut height = 350)	0.209615	11347.575582
Fuzzy CMeans(K=2)	0.204777	10897.056752
Hierarchical (dendrogram cut height = 250)	0.176792	9618.795831
Agglomerative (K=3)	0.176832	9618.795831
KMeans(K=4)	0.224022	9022.603463
Gaussian Mixture (K=4)	0.218829	9008.029901
Hierarchical (dendrogram cut height = 220)	0.197746	8538.205959
Bisecting KMeans (K=5)	0.23132	8174.36548
Birch (K=4)	0.19966	8112.911935
Hierarchical (dendrogram cut height = 175)	0.248036	7675.246798
Birch (K=4, important cols)	0.191999	6783.58443
Gaussian Mixture (K=8)	0.236881	6172.480164
KMeans(K=5, standardization)	0.169814	4400.753985
Spectral (K=3)	-0.061996	16.677706

In [44]:
```
kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit_predict(df)
pca = PCA(n_components=5)
X_pca = pca.fit_transform(df)
plot_2d(X_pca, clusters)
fi = plot_feature_importance(df, clusters)
```





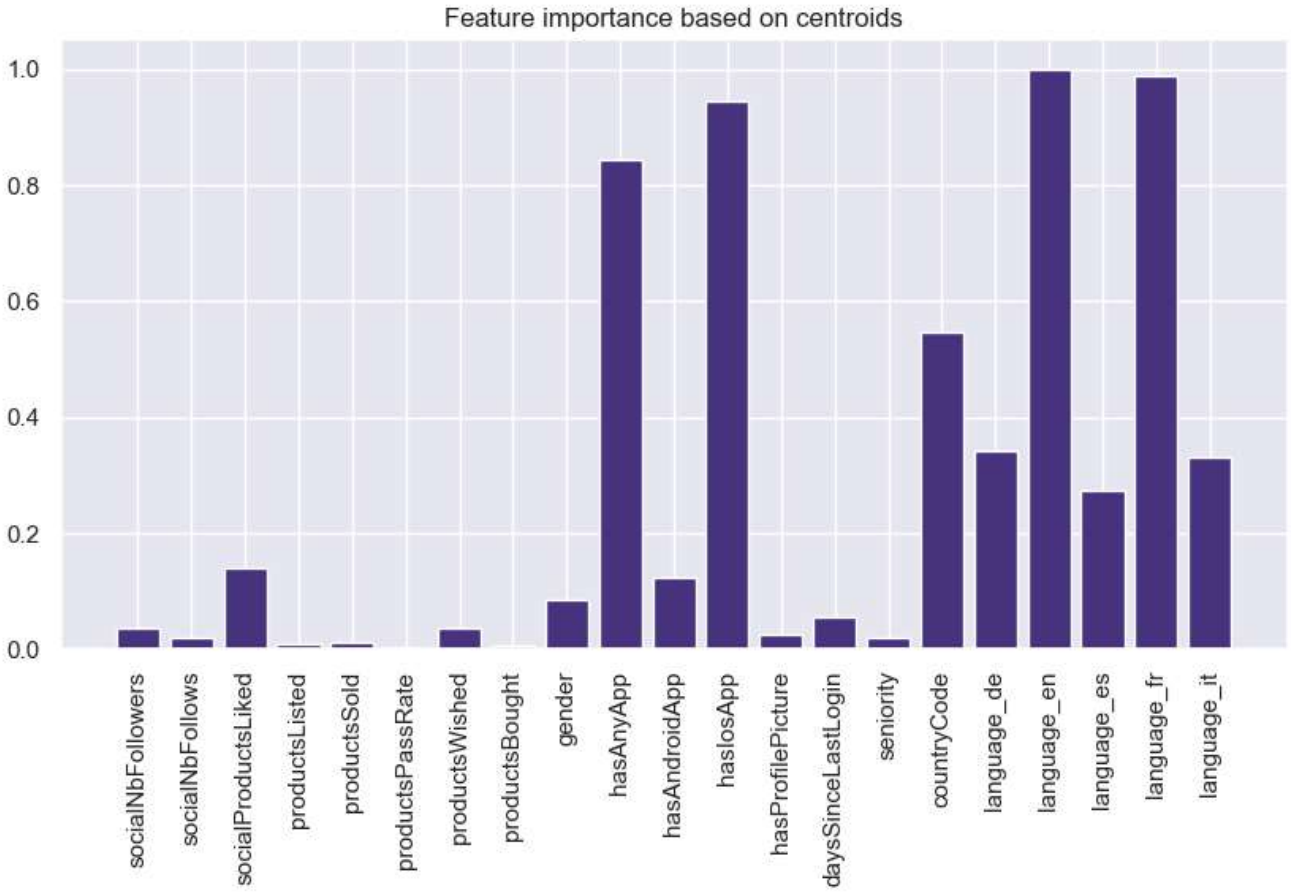
In [45]:
```
kmeans = KMeans(n_clusters=4, random_state=42)
df_features = df.drop(columns=df.columns[fi < 0.2])
clusters = kmeans.fit_predict(df_features)
plot_2d(X_pca, clusters)
```



In [46]:
```
best_clusters = clusters
```

Analiza klastera i poslovna strategija

In [47]:
```
fi = plot_feature_importance(df, best_clusters)
analyze_features(df, fi, best_clusters, 4)
```



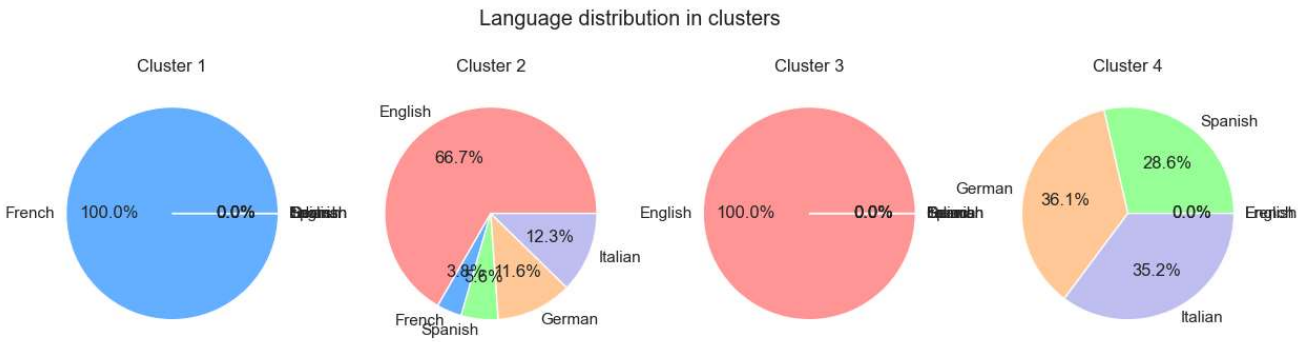
Out[47]:

	0	1	2	3
hasAnyApp	-0.247730	1.000000	-0.870212	-0.691414
hasIosApp	-0.448018	1.000000	-1.000000	-1.000000
countryCode	0.759195	-0.399938	-0.244558	-0.331502
language_de	-1.000000	-0.768405	-1.000000	-0.277202
language_en	-1.000000	0.333742	1.000000	-1.000000
language_es	-1.000000	-0.887270	-1.000000	-0.427347
language_fr	1.000000	-0.923773	-1.000000	-1.000000
language_it	-1.000000	-0.754294	-1.000000	-0.295451

In [48]:
```
clusters = [df[best_clusters == i] for i in range(4)]
```

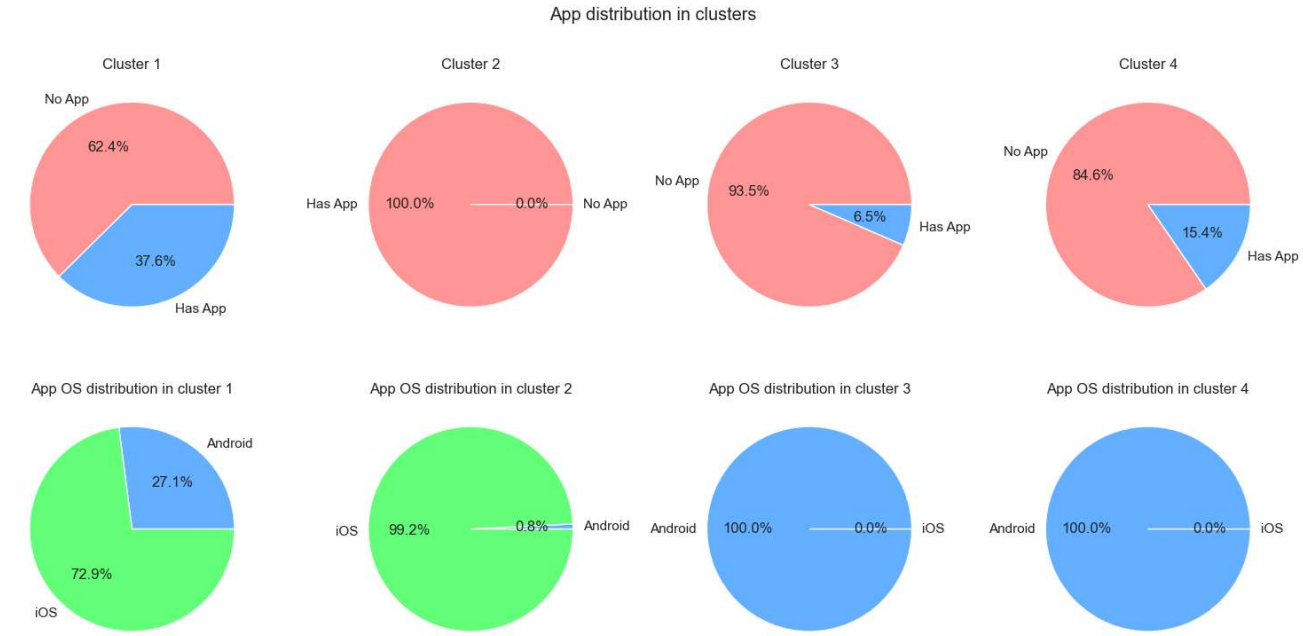
In [49]:
```
plt.figure(figsize=(15, 4))
for i, cluster in enumerate(clusters):
    plt.subplot(1, 4, i+1)
    cluster = pd.DataFrame(
        scaler.inverse_transform(cluster), columns=df.columns)
    language_counts = cluster['language_en'].sum(), \
        cluster['language_fr'].sum(), \
        cluster['language_es'].sum(), \
        cluster['language_de'].sum(), \
        cluster['language_it'].sum()
    language_labels = ['English', 'French', 'Spanish', 'German', 'Italian']
    plt.pie(language_counts, labels=language_labels,
            autopct='%1.1f%%', colors=[
                '#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#c2c2f0'])
```

```
    plt.title(f'Cluster {i+1}')
plt.suptitle('Language distribution in clusters')
plt.show()
```

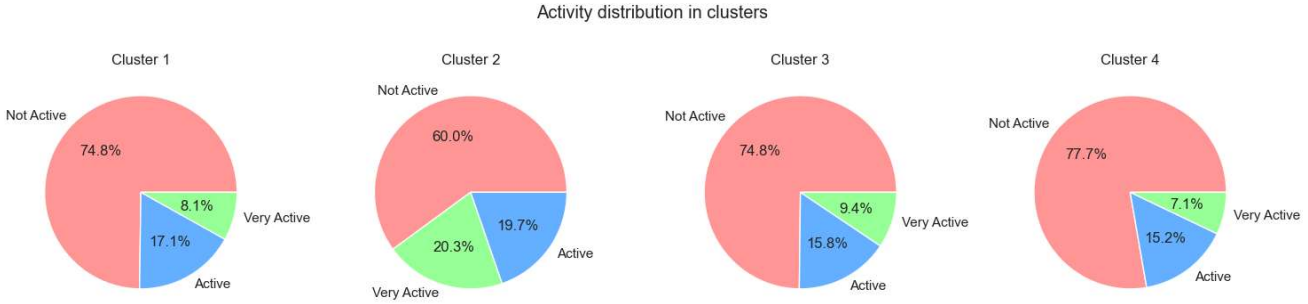


In [59]:

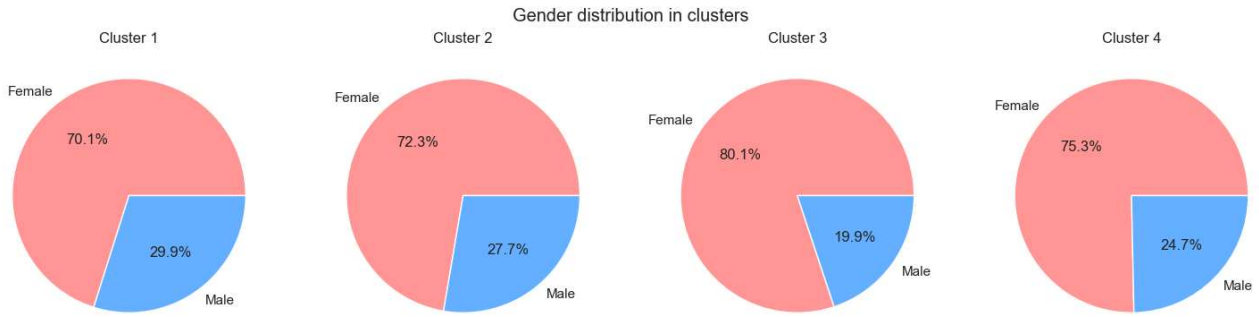
```
plt.figure(figsize=(15, 8))
for i, cluster in enumerate(clusters):
    plt.subplot(2, 4, i+1)
    cluster = pd.DataFrame(
        scaler.inverse_transform(cluster), columns=df.columns)
    has_app_counts = cluster['hasAnyApp'].value_counts()
    if 0 not in has_app_counts.index:
        has_app_counts[0] = 0
    label_mapping = {0: 'No App', 1: 'Has App'}
    labels = [label_mapping[i]
              for i in has_app_counts.index]
    plt.pie(has_app_counts, labels=labels, autopct='%1.1f%%', colors=[
        '#ff9999', '#66b3ff'])
    plt.title(f'Cluster {i+1}')

    plt.subplot(2, 4, 4+i+1)

    has_app_counts = cluster['hasAndroidApp'].sum(), cluster['hasIosApp'].sum()
    plt.pie(has_app_counts, labels=['Android', 'iOS'], autopct='%1.1f%%', colors=[
        '#66b3ff', '#66ff7c'])
    plt.title(f'App OS distribution in cluster {i+1}')

plt.suptitle('App distribution in clusters')
plt.tight_layout()
plt.show()
```



In [62]:
```
plt.figure(figsize=(15, 4))
for i, cluster in enumerate(clusters):
    plt.subplot(1, 4, i+1)
    cluster = pd.DataFrame(
        scaler.inverse_transform(cluster), columns=df.columns)
    has_app_counts = cluster['socialProductsLiked'].value_counts()
    if 0 not in has_app_counts.index:
        has_app_counts[0] = 0
    label_mapping = {0: 'Not Active', 1: 'Active', 2: 'Very Active'}
    labels = [label_mapping[i]
              for i in has_app_counts.index]
    color_mapping = {0: '#ff9999', 1: '#66b3ff', 2: '#99ff99'}
    colors = [color_mapping[i]
              for i in has_app_counts.index]
    plt.pie(has_app_counts, labels=labels, autopct='%1.1f%%', colors=colors)
    plt.title(f'Cluster {i+1}')


plt.suptitle('Activity distribution in clusters')
plt.tight_layout()
plt.show()
```

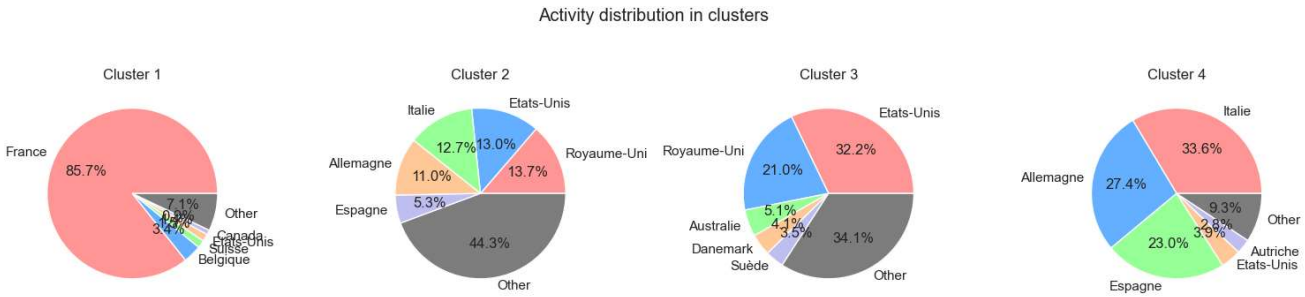


In [60]:
```
plt.figure(figsize=(15, 4))
for i, cluster in enumerate(clusters):
    plt.subplot(1, 4, i+1)
    cluster = pd.DataFrame(
        scaler.inverse_transform(cluster), columns=df.columns)
    has_app_counts = cluster['gender'].value_counts()
    if 0 not in has_app_counts.index:
        has_app_counts[0] = 0
    label_mapping = {0: 'Male', 1: 'Female'}
    labels = [label_mapping[i]
              for i in has_app_counts.index]
    plt.pie(has_app_counts, labels=labels, autopct='%1.1f%%', colors=[
        '#ff9999', '#66b3ff'])
    plt.title(f'Cluster {i+1}')


plt.suptitle('Gender distribution in clusters')
plt.tight_layout()
plt.show()
```



In [84]:
```
count_to_country = {v: k for k, v in country_count.items()}
print(count_to_country)

plt.figure(figsize=(15, 4))
for i, cluster in enumerate(clusters):
    plt.subplot(1, 4, i+1)
    cluster = pd.DataFrame(
        scaler.inverse_transform(cluster), columns=df.columns)
    country_counts = cluster['countryCode'].value_counts()
    top_countries = country_counts.iloc[:5]
    other_countries = country_counts.iloc[5:]
    other_count = other_countries.sum()
    top_countries['Other'] = other_count
    labels = []
    for l in top_countries.index:
        if l == 'Other':
            labels.append('Other')
        else:
            labels.append(count_to_country[round(l)])
    colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#c2c2f0', '#808080']
    plt.pie(top_countries, labels=labels, autopct='%1.1f%%', colors=colors)
    plt.title(f'Cluster {i+1}')


plt.suptitle('Country distribution in clusters')
plt.tight_layout()
plt.show()
```

```
{11145: 'France', 7459: 'Etats-Unis', 5454: 'Royaume-Uni', 4628: 'Italie', 3836: 'Allemagn
e', 2720: 'Espagne', 1527: 'Australie', 1278: 'Danemark', 1193: 'Suède', 1047: 'Belgique',
978: 'Pays-Bas', 803: 'Canada', 609: 'Suisse', 526: 'Hong Kong', 460: 'Finlande', 454: 'Au
triche', 287: 'Chine', 284: 'Irlande', 258: 'Roumanie', 253: 'Russie', 178: 'Portugal', 16
3: 'Grèce', 160: 'Pologne', 138: 'Norvège', 127: 'Japon', 124: 'Mexique', 112: 'Maroc', 10
9: 'Inde', 101: 'Taiwan', 99: 'Luxembourg', 96: 'Corée du Sud', 95: 'Émirats arabes unis',
94: 'Croatie', 93: 'Arabie Saoudite', 91: 'Malaisie', 82: 'Ukraine', 81: 'Israel', 78: 'No
uvelle Zélande', 76: 'Bulgarie', 72: 'Philippines', 69: 'Tunisie', 66: 'Afrique du Sud', 6
5: 'Brésil', 63: 'Algérie', 62: 'Thailande', 61: 'Kowait', 59: 'Indonésie', 56: 'Hongrie',
52: 'République tchèque', 48: 'Chypre', 46: 'Slovaquie', 45: 'Lettonie', 44: 'Chili', 42:
'Qatar', 41: 'Serbie', 37: 'Lituanie', 35: 'Guadeloupe', 34: 'Monaco', 32: 'Estonie', 31:
'Kazakhstan', 28: "Cote D'Ivoire", 27: 'Martinique', 25: 'Nigeria', 24: 'Bahreïn', 23: 'Eg
ypte', 22: 'Géorgie', 20: 'Azerbaïdjan', 19: 'Cameroun', 18: 'Puerto Rico', 16: 'Congo', 1
5: 'Macau', 14: 'Bosnie et Herzegovine', 13: 'Équateur', 12: 'Nouvelle Calédonie', 11: 'Ke
nya', 10: 'Jersey', 9: 'Bangladesh', 8: 'Costa Rica', 7: 'Bermude', 6: 'Honduras', 5: 'Tri
nidad et Tobago', 4: 'Kirghizistan', 3: 'Îles Turques-et-Caïques ', 2: 'Botswana', 1: 'Île
de Man'}
```



Analiza klastera

Klaster 1:

- korisnici koji govore francuski
- trećina poseduje aplikaciju, uglavnom iOS
- četvrtina njih je aktivno

Klaster 2:

- pretežno govore engleski, a po 12% italijanski i nemački
- imaju iOS aplikaciju
- aktivniji korisnici u ovom klasteru (trećina ih je aktivno)

Klaster 3:

- korisnici koji govore engleski
- većinski nemaju aplikaciju (oni koji imaju, poseduju Android aplikaciju)
- četvrtina njih je aktivno

Klaster 4:

- korisnici koji govore italijanski, nemački i španski
- uglavnom nemaju aplikaciju (oni koji imaju poseduju Andorid aplikaciju)
- najmanje aktivni (nešto manje od četvrtine)

Poslovna strategija

1. Nuditi popuste i kupone, prikazivati reklame i slati obaveštenja korisnicima koji poseduju iOS aplikaciju, jer su oni najaktivniji
2. Korisnicima koji nemaju aplikaciju slati ponude na e-mail kako bi se promovisala stranica
3. Promovisati aplikaciju ili ponude korisnicima sa ostalih govornih područja (francuski, italijanski, nemački španski), koji su manje aktivni od korisnika sa engleskog govornog područja
    - Razmotriti implementaciju naprednijih metoda za prevođenje kako bi se poboljšalo interesovanje i komunikacija sa klijentima sa ostalih govornih područja