3RCFU (Three Rivers Credit Federal Unit Data Analysis - Project Part II)

Census Neighborhood Metrics Table (First 33 Variables)

Path for the all data files requied to run the code:

/anvil/projects/x-cis220051/corporate/3rivers-membership/projects/x-abajpayee

In [2]:
```
import pandas as pd
import numpy as np
import os
import pickle
import shutil
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import seaborn as sns
import pyreadr
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
from tqdm import tqdm
%matplotlib inline
```

In [3]:
```
census = pyreadr.read_r('Data' + '/Census_Neighborhood_Metrics_Table.rds')[N
print(census.shape)
census.head()
```

```
(22230, 48)
```

Out[3]:

	censuscode	Percent_Individual_Income_lt10K	Percent_Individual_Income_10to15K	Percer
0	GKA4709381	0.171642	0.210199	
1	XPG8453176	NaN	NaN	
2	ILR6895472	0.123532	0.062636	
3	JAM2038971	0.171086	0.079555	
4	GTE6315027	0.222871	0.121566	

5 rows × 48 columns

In [4]:
```
census.describe()
```

Out[4]:

	Percent_Individual_Income_lt10K	Percent_Individual_Income_10to15K	Percent_Individu
count	19215.000000	19215.000000	
mean	0.135743	0.076392	
std	0.055477	0.035296	
min	0.000000	0.000000	
25%	0.101804	0.051340	
50%	0.127421	0.071639	
75%	0.158771	0.096345	
max	0.789583	0.380843	

8 rows × 47 columns

In [5]:

```
census.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22230 entries, 0 to 22229
Data columns (total 48 columns):
 #   Column                                        Non-Null Count  Dtype  
---  ------                                        --------------  -----  
 0   censuscode                                    22230 non-null  object 
 1   Percent_Individual_Income_lt10K               19215 non-null  float64
 2   Percent_Individual_Income_10to15K             19215 non-null  float64
 3   Percent_Individual_Income_15to25K             19215 non-null  float64
 4   Percent_Individual_Income_25to35K             19215 non-null  float64
 5   Percent_Individual_Income_35to50K             19215 non-null  float64
 6   Percent_Individual_Income_50to65K             19215 non-null  float64
 7   Percent_Individual_Income_65to75K             19215 non-null  float64
 8   Percent_Individual_Income_gte75K              19215 non-null  float64
 9   Percent_Poverty                               19213 non-null  float64
 10  Percent_neverMarried                          19215 non-null  float64
 11  Percent_Married                               19215 non-null  float64
 12  Percent_Education_HSgrad                      19214 non-null  float64
 13  Percent_Education_Somecollegeorassociate      19214 non-null  float64
 14  Percent_Education_Bachelor                    19214 non-null  float64
 15  Percent_Education_Graduateorprofessionaldegree 19214 non-null  float64
 16  Percent_FoodStamps_Household                  19214 non-null  float64
 17  Percent_GovAsst_Child_Household_SSI_SNAP_CPAI 19123 non-null  float64
 18  Percent_Unemployed                            19214 non-null  float64
 19  Percent_Family_Poverty                        19210 non-null  float64
 20  Percent_Medicaid                              19215 non-null  float64
 21  Percent_HomeOwner                             19218 non-null  float64
 22  Percent_Foreign_Born                          19219 non-null  float64
 23  Percent_JobSector_Gov                         19218 non-null  float64
 24  Percent_JobSector_SelfEmploy                  19218 non-null  float64
 25  Population_Density                            19168 non-null  float64
 26  Percent_Black                                 19167 non-null  float64
```

```
 27  Percent_Native_American                        19167 non-null  float64
 28  Percent_Asian                                  19167 non-null  float64
 29  Percent_Pacific_Islander                       19167 non-null  float64
 30  Percent_Other                                  19167 non-null  float64
 31  Percent_gteTwoRaces                            19167 non-null  float64
 32  Percent_Hispanic                               19167 non-null  float64
 33  Percent_Age_lt18                               19216 non-null  float64
 34  Percent_Age_18to24                             19216 non-null  float64
 35  Percent_Age_gte65                              19216 non-null  float64
 36  Income_Median                                  19163 non-null  float64
 37  Household_Income_Mean_Lowest_Quintile          18046 non-null  float64
 38  Household_Income_Mean_Second_Quintile          18046 non-null  float64
 39  Household_Income_Mean_Third_Quintile           18046 non-null  float64
 40  Household_Income_Mean_Fourth_Quintile          18046 non-null  float64
 41  Household_Income_Mean_Highest_Quintile         18046 non-null  float64
 42  GINI_Index                                     19174 non-null  float64
 43  Household_Income_Median                        19145 non-null  float64
 44  GrossRent_Median                               18286 non-null  float64
 45  HousingUnit_Value_Median                       18874 non-null  float64
 46  RealEstate_Taxes_Median                        18833 non-null  float64
 47  MonthHousing_Costs_Median                      19162 non-null  float64
dtypes: float64(47), object(1)
memory usage: 8.1+ MB
```

Analysis to find null and missing values

In [6]:

```
100*(census.Percent_Individual_Income_lt10K.isna().sum()/census.shape[0])
```

Out[6]:

```
13.562753036437247
```

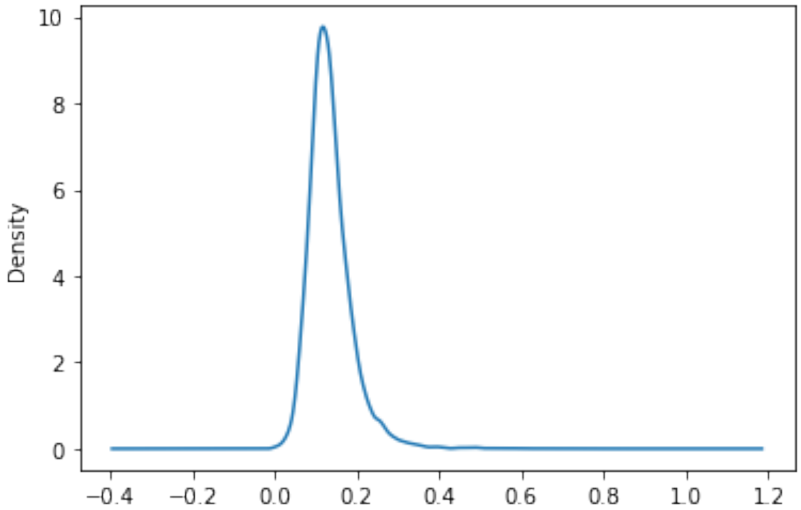
Around 13% of the values for payoffdate are missing this might be because these 28% of customers haven't paid off the loan yet. So, we can convert this feature into binary values feature, turning all missing values into '1' and the rest in '0', where '1' indicates that the loan is yet to be paid off and '0' indicates that the loan has been paid of The traditional means of imputing missing value like mean, median and mode won't really be helpful so we might not include this feature

In [7]:

```
census.Percent_Individual_Income_lt10K.plot(kind='kde')
```

Out[7]:

```
<AxesSubplot:ylabel='Density'>
```



In [8]:
```
census.Percent_Individual_Income_lt10K.describe()
```

Out[8]:
```
count    19215.000000
mean         0.135743
std          0.055477
min          0.000000
25%          0.101804
50%          0.127421
75%          0.158771
max          0.789583
Name: Percent_Individual_Income_lt10K, dtype: float64
```

Inplacing those missing values with median of the column

In [9]:
```
census.Percent_Individual_Income_lt10K.fillna( census.Percent_Individual_Inc
census.Percent_Individual_Income_15to25K.fillna( census.Percent_Individual_I
census.Percent_Individual_Income_25to35K.fillna( census.Percent_Individual_I
census.Percent_Individual_Income_35to50K.fillna( census.Percent_Individual_I
census.Percent_Individual_Income_50to65K.fillna( census.Percent_Individual_I
census.Percent_Individual_Income_65to75K.fillna( census.Percent_Individual_I
census.Percent_Individual_Income_gte75K.fillna( census.Percent_Individual_In
census.Percent_Poverty.fillna( census.Percent_Poverty.median(), inplace=True
census.Percent_neverMarried.fillna( census.Percent_neverMarried.median(), in
census.Percent_Married.fillna( census.Percent_Married.median(), inplace=True
census.Percent_Education_HSgrad.fillna( census.Percent_Education_HSgrad.medi
census.Percent_Education_Somecollegeorassociate.fillna( census.Percent_Educa
census.Percent_Education_Bachelor.fillna( census.Percent_Education_Bachelor.
census.Percent_Education_Graduateorprofessionaldegree.fillna( census.Percent
census.Percent_FoodStamps_Household.fillna( census.Percent_FoodStamps_Househ
census.Percent_GovAsst_Child_Household_SSI_SNAP_CPAI.fillna( census.Percent_
census.Percent_Unemployed.fillna( census.Percent_Unemployed.median(), inplac
census.Percent_Family_Poverty.fillna( census.Percent_Family_Poverty.median()
census.Percent_Medicaid.fillna( census.Percent_Medicaid.median(), inplace=Tr
census.Percent_HomeOwner.fillna( census.Percent_HomeOwner.median(), inplace=
census.Percent_Foreign_Born.fillna( census.Percent_Foreign_Born.median(), in
census.Percent_JobSector_Gov.fillna( census.Percent_JobSector_Gov.median(),
census.Percent_JobSector_SelfEmploy.fillna( census.Percent_JobSector_SelfEmp
census.Population_Density.fillna( census.Population_Density.median(), inplac
census.Percent_Black.fillna( census.Percent_Black.median(), inplace=True )
census.Percent_Native_American.fillna( census.Percent_Native_American.median
census.Percent_Asian.fillna( census.Percent_Asian.median(), inplace=True )
census.Percent_Pacific_Islander.fillna( census.Percent_Pacific_Islander.medi
census.Percent_Other.fillna( census.Percent_Other.median(), inplace=True )
census.Percent_gteTwoRaces.fillna( census.Percent_gteTwoRaces.median(), inpl
census.Percent_Hispanic.fillna( census.Percent_Hispanic.median(), inplace=Tr
```

Observation (Population Density):

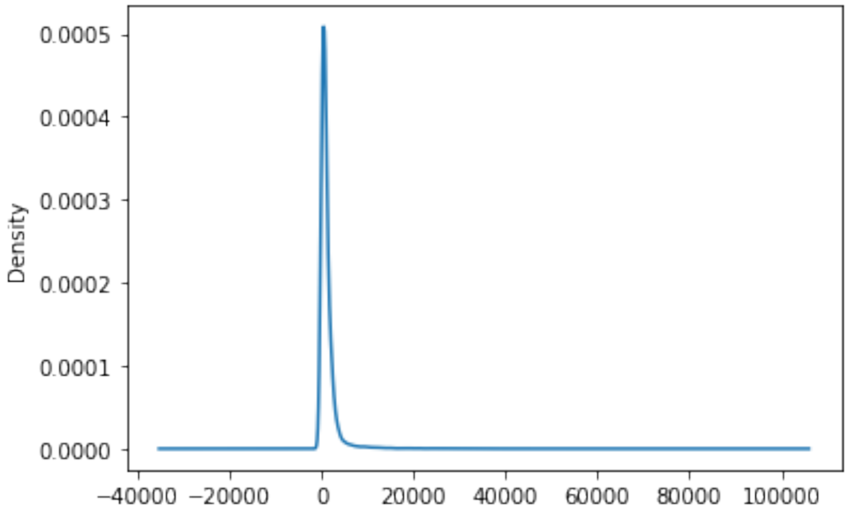
census.Population_Density.describe()

In [13]:
```
100*(census.Population_Density.isna().sum()/census.shape[0])
```

Out[13]:
```
0.0
```

In [14]:
```
census.Population_Density.plot(kind='kde')
```

Out[14]:
```
<AxesSubplot:ylabel='Density'>
```



In [15]:
```
census.Population_Density.describe()
```

Out[15]:
```
count    22230.000000
mean      1311.766802
std       2963.941214
min          0.000000
25%        229.309446
50%        727.320863
75%       1384.060317
max      70640.290323
Name: Population_Density, dtype: float64
```

The statistical summary for the Population_Density variable in the dataset presents a distinct profile compared to the other 32 columns, which are primarily composed of percentage values related to various metrics. Unlike these percentage-based columns, the Population_Density is measured in absolute terms, providing concrete figures rather than relative percentages.

In this dataset, the Population_Density variable has a total of 22,230 entries, indicating a substantial data size. The average population density, denoted by the mean, is approximately 1,311.77. This figure represents the average number of individuals per unit area across all the measured locations. However, the standard deviation is significantly high at around 2,963.94, revealing a broad dispersion of population density values. This wide range suggests that the dataset encompasses a diverse range of areas, from sparsely populated to highly urbanized regions.

The minimum value recorded for population density is 0, which might indicate areas with negligible or no population. The 25th percentile is at 229.31, meaning a quarter of the areas have this density or lower, pointing to a number of less populated regions. The median value, or the 50th percentile, is 727.32, which is markedly lower than the mean, suggesting a right-skewed distribution. This skewness indicates that while most areas have a moderate population density, there are a few areas with extremely high densities that increase the average.

Finally, the maximum value in the dataset for population density is an exceptionally high 70,640.29, confirming the presence of highly urbanized areas within the dataset. This extreme value contrasts sharply with the overall moderate densities observed in most areas, further emphasizing the diverse range of environments covered in the dataset.

In [16]:
```
normalization_features = [ 'Percent_Individual_Income_lt10K','Percent_Indivi
```

In [17]:
```
updated_normalization_features = [ 'Percent_Individual_Income_lt10K','Percen
```

Normalization of the 2 lists

In [19]:
```
scaler = MinMaxScaler()
scaler.fit(census[normalization_features])
scaler.fit(census[updated_normalization_features])
```

Out[19]:
```
▾ MinMaxScaler
MinMaxScaler()
```

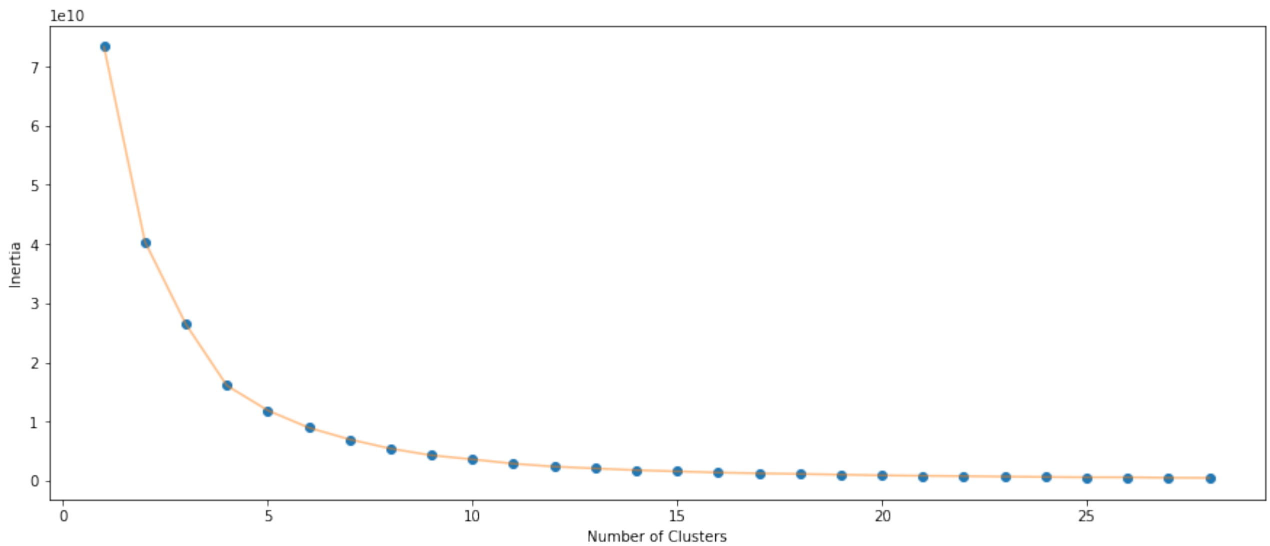
In [25]:
```
normalized_census = census[normalization_features]
updated_normalized_census = census[updated_normalization_features]
```

Determining the optimal number of clusters using elbow method

In [26]:
```
inertia = []
for n in tqdm(range(2 , 30)):
    algorithm = (KMeans(n_clusters = n ,init='k-means++', n_init = 3 ,max_it
                        tol=0.0001,  random_state= 111  , algorithm='elkan')
    algorithm.fit(normalized_census)
    inertia.append(algorithm.inertia_)

plt.figure(1 , figsize = (15 ,6))
plt.plot(np.arange(1 , 29) , inertia , 'o')
plt.plot(np.arange(1 , 29) , inertia , '-' , alpha = 0.5)
plt.xlabel('Number of Clusters') , plt.ylabel('Inertia')
plt.show()
```

```
100%|██████████| 28/28 [00:18<00:00,  1.48it/s]
```

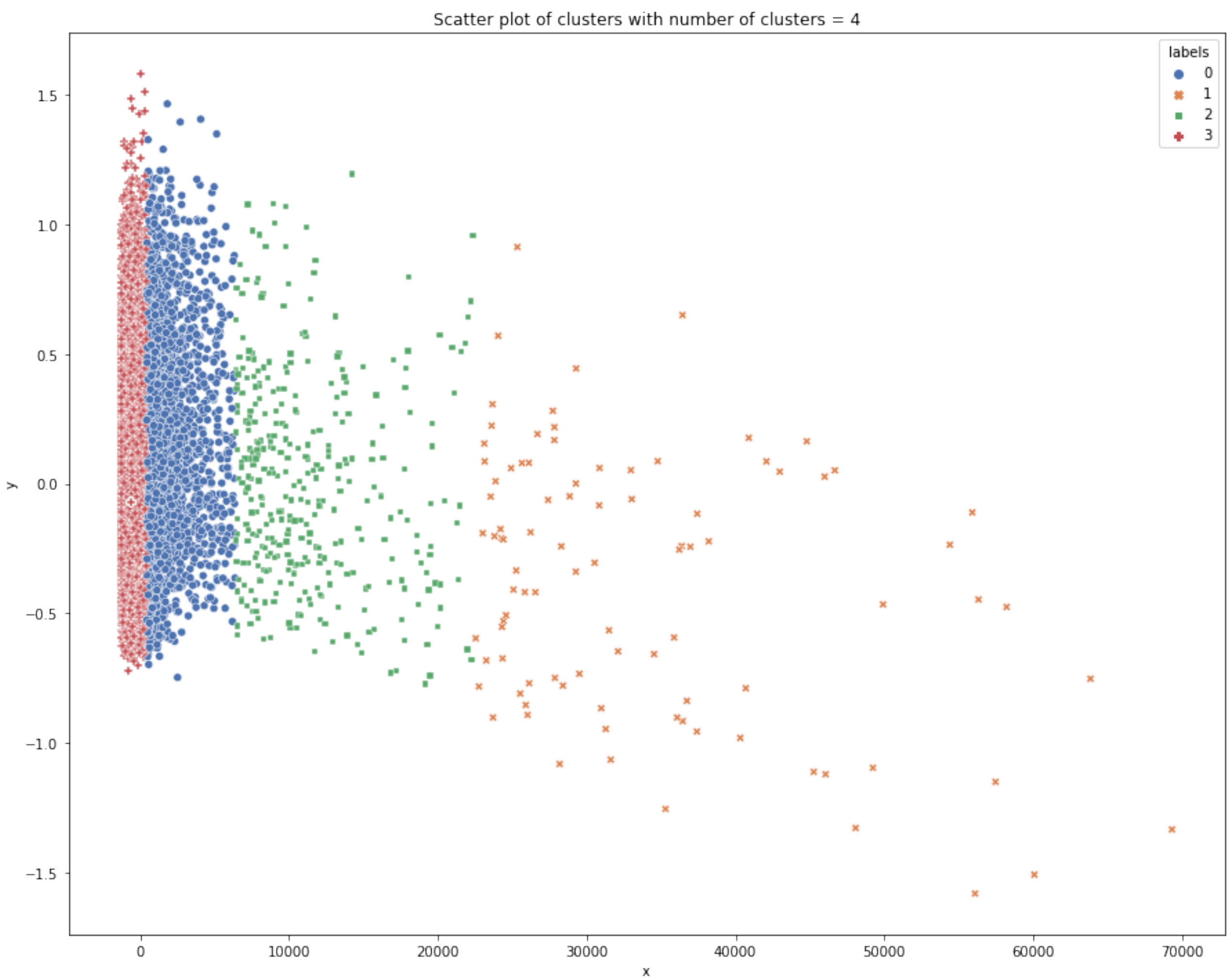


Clustering plots:

In [23]:
```
algorithm = (KMeans(n_clusters = 4 ,init='k-means++', n_init = 3 ,max_iter=5
                        tol=0.0001,  random_state= 111  , algorithm='elkan') )
algorithm.fit(normalized_census)
labels1 = algorithm.labels_
centroids1 = algorithm.cluster_centers_

pca = PCA(n_components=2)
pca.fit(normalized_census)
reduced_data = pca.transform(normalized_census)
plt.figure(1, figsize=(15,12))
visual_df = pd.DataFrame( {'x' : reduced_data[:,0], 'y' : reduced_data[:,1],
sns.scatterplot(data=visual_df, x='x', y='y', hue='labels', style='labels',
plt.title('Scatter plot of clusters with number of clusters = 4')
```

Out[23]:
```
Text(0.5, 1.0, 'Scatter plot of clusters with number of clusters = 4')
```

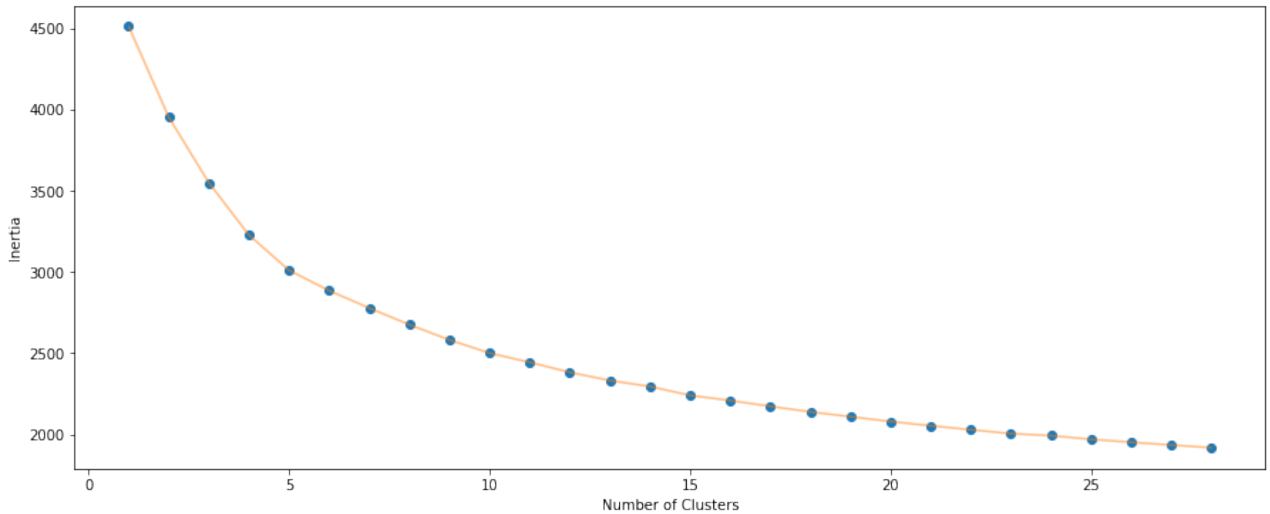


Determining the optimal number of clusters using elbow method excluding the column, "Population Density"

In [22]:
```
inertia = []
for n in tqdm(range(2 , 30)):
    algorithm = (KMeans(n_clusters = n ,init='k-means++', n_init = 3 ,max_it
                        tol=0.0001,  random_state= 111  , algorithm='elkan')
    algorithm.fit(updated_normalized_census)
    inertia.append(algorithm.inertia_)

plt.figure(1 , figsize = (15 ,6))
plt.plot(np.arange(1 , 29) , inertia , 'o')
plt.plot(np.arange(1 , 29) , inertia , '-' , alpha = 0.5)
plt.xlabel('Number of Clusters') , plt.ylabel('Inertia')
plt.show()
```

```
100%|██████████| 28/28 [00:26<00:00,  1.05it/s]
```

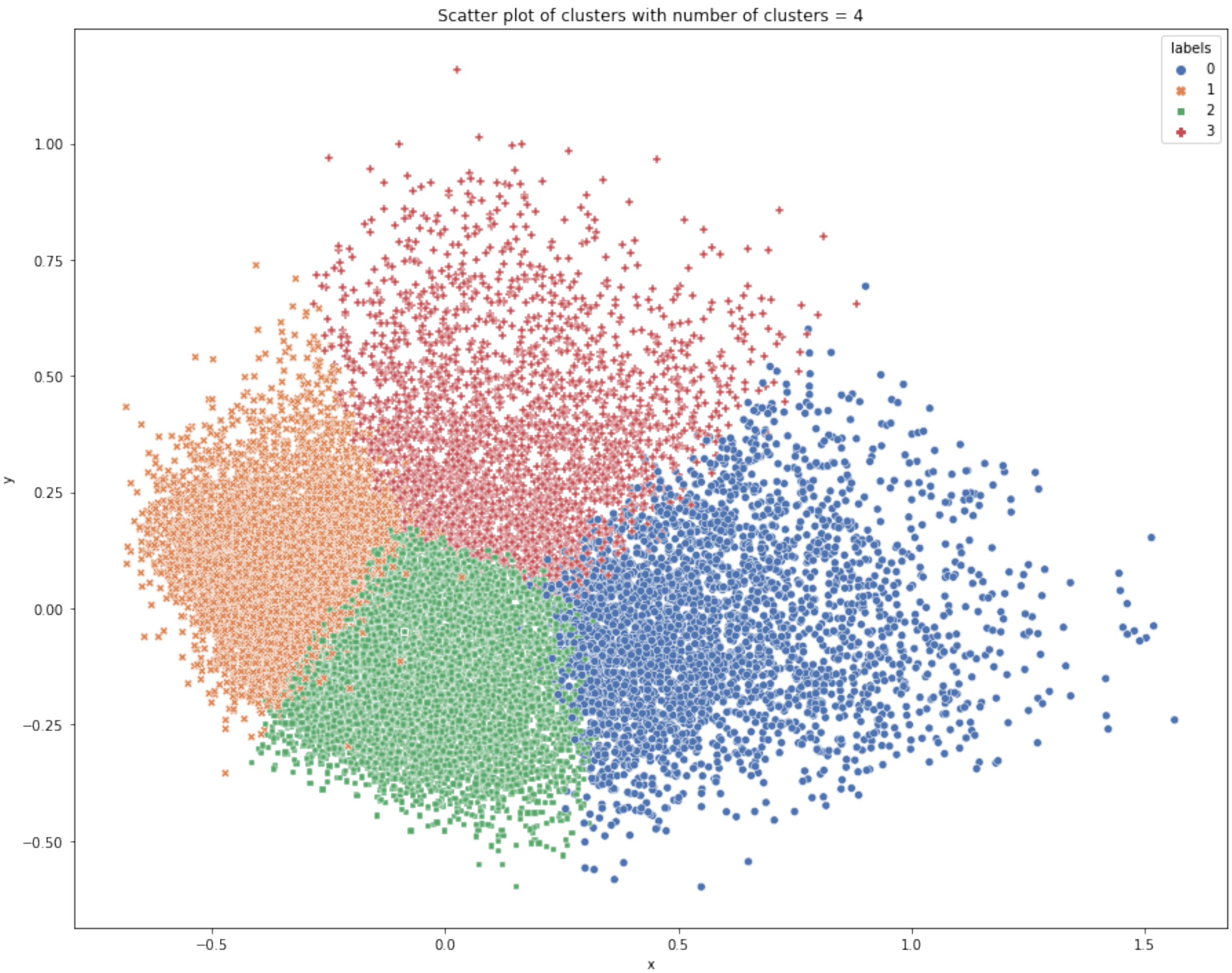


Clustering plots:

In [24]:
```
algorithm = (KMeans(n_clusters = 4 ,init='k-means++', n_init = 3 ,max_iter=5
                    tol=0.0001,  random_state= 111  , algorithm='elkan') )
algorithm.fit(updated_normalized_census)
labels2 = algorithm.labels_
centroids2 = algorithm.cluster_centers_

pca = PCA(n_components=2)
pca.fit(updated_normalized_census)
reduced_data = pca.transform(updated_normalized_census)
plt.figure(1, figsize=(15,12))
visual_df = pd.DataFrame( {'x' : reduced_data[:,0], 'y' : reduced_data[:,1],
sns.scatterplot(data=visual_df, x='x', y='y', hue='labels', style='labels',
plt.title('Scatter plot of clusters with number of clusters = 4')
```

Out[24]:
```
Text(0.5, 1.0, 'Scatter plot of clusters with number of clusters = 4')
```



Saving the data and corresponding cluster labels

In [76]:
```
solution_df = pd.DataFrame(
    {
    'censuscode' : census['censuscode'],
    'labels' : labels2
    }
)
solution_df.to_csv("census_labelled.csv", index=None)
np.unique(solution_df.labels)
```

Out[76]: array([0, 1, 2, 3], dtype=int32)

Summary Report on K-Means++ Clustering Analysis of Census Data

Dataset Overview: The dataset comprises 22,230 entries, each representing a unique census code. It contains 33 columns, covering a wide range of socio-economic and demographic indicators, such as individual income brackets, marital status, educational attainment, employment status, racial demographics, and more. Clustering Analysis:

Objective:

To segment the dataset into four distinct clusters using the k-means++ algorithm.

Methodology:

With 'Population Density': The initial clustering included all columns, notably 'Population Density'. Without 'Population Density': A second clustering analysis was conducted after removing the 'Population Density' column. Findings: Clusters with 'Population Density': The inclusion of 'Population Density' in the analysis led to clusters where this variable significantly influenced the grouping. This might have overshadowed other socio-economic factors due to the high variance or distinct patterns in population density across regions. Clusters without 'Population Density': Removing 'Population Density' resulted in clusters more reflective of socio-economic and demographic similarities, not overshadowed by geographical density factors. This approach likely offered a clearer view of how other variables, such as income levels, educational attainment, and racial composition, interact and cluster independently of geographical density considerations. Implications and Recommendations:

Comparative Insights:

The comparison between the two clustering results highlights how a single variable, like 'Population Density', can significantly skew or influence cluster formation. Decision for Analysis: The clusters formed without 'Population Density' appear to be more informative for socio-economic and demographic analysis, suggesting its exclusion is beneficial for certain types of analysis. Future Studies: It's recommended to consider the specific research goals when deciding whether to include or exclude variables like 'Population Density'. For studies focusing on urban-rural divides or regional planning, including it might be more relevant.

Conclusion:

The k-means++ clustering analysis, with and without the 'Population Density' variable, provides valuable insights into how different variables influence data segmentation. The findings underscore the importance of careful variable selection in clustering algorithms to yield the most meaningful and relevant results for the intended analysis.