In [1]:
```
#Razat Siwakoti (A00046635)
#DMV302 - Assessment 2
#Kmeans1.ipynb created on Jupyter notebook
```

In [2]:
```
#source code: CihanBosnali (2019)
#https://github.com/CihanBosnali/Machine-Learning-without-Libraries/blob/master/K-Means-

#Prasanth S N (2020)
#https://ai538393399.wordpress.com/2020/09/29/k-means-clustering-algorithm-without-libra
```

In [3]:
```
# Importing necessary libraries
import os
os.getcwd()

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px

# Setting the plotting style to 'dark_background'
plt.style.use('dark_background')
```

In [4]:
```
# Reading the CSV file "HouseholdWealth.csv" into a pandas DataFrame
df = pd.read_csv("HouseholdWealth.csv")
# Displaying the first few rows of the DataFrame to get an overview
df.head()
```

Out[4]:

	household_total_assets	annual_household_income
0	1230531	15724
1	4877446	124751
2	4430878	124372
3	1954751	179311
4	2179963	56355

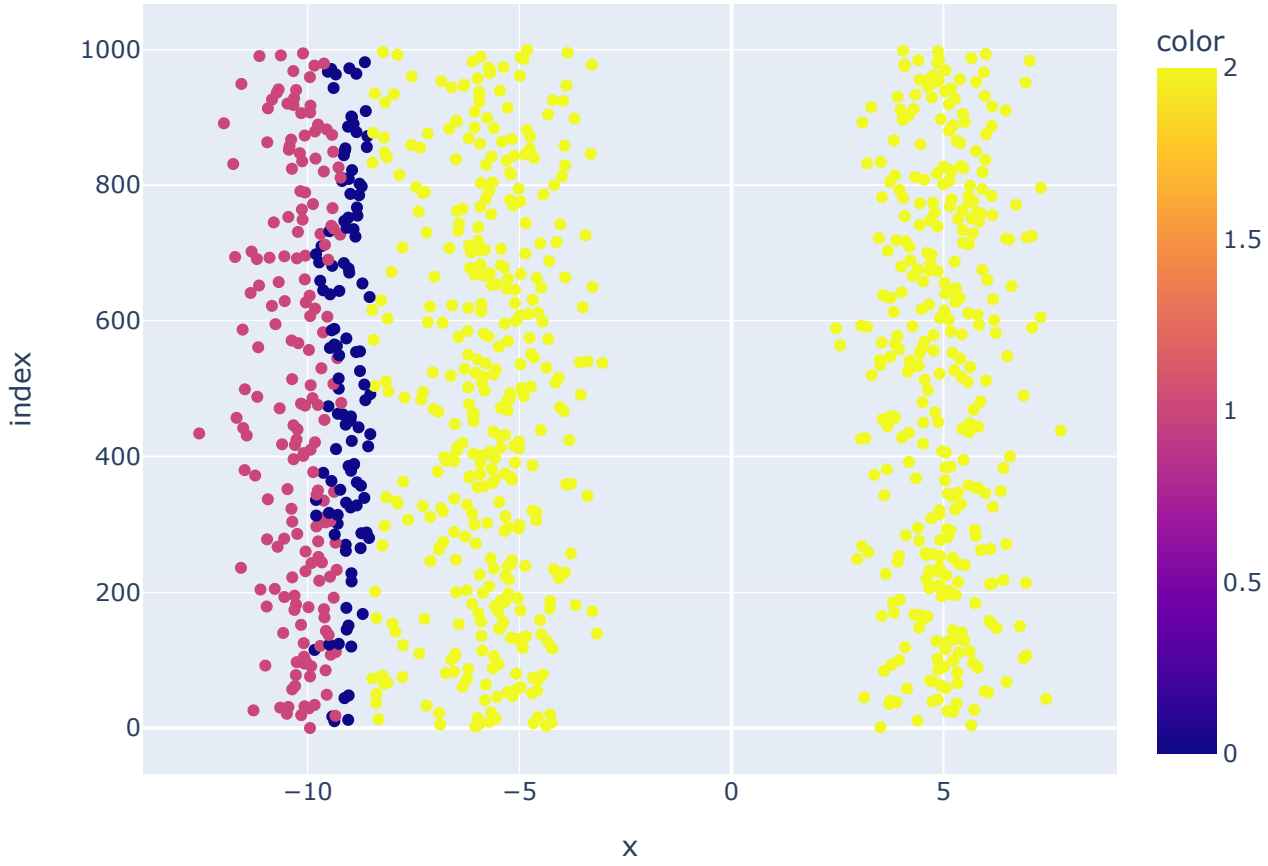
In [5]:
```
k = 3  # Setting the number of clusters to k
# Initializing a dictionary to store clusters, where keys are cluster indices and values
clusters = {}
# Looping through the range of k to create empty lists for each cluster
for i in range(k):
    clusters[i] = []
```

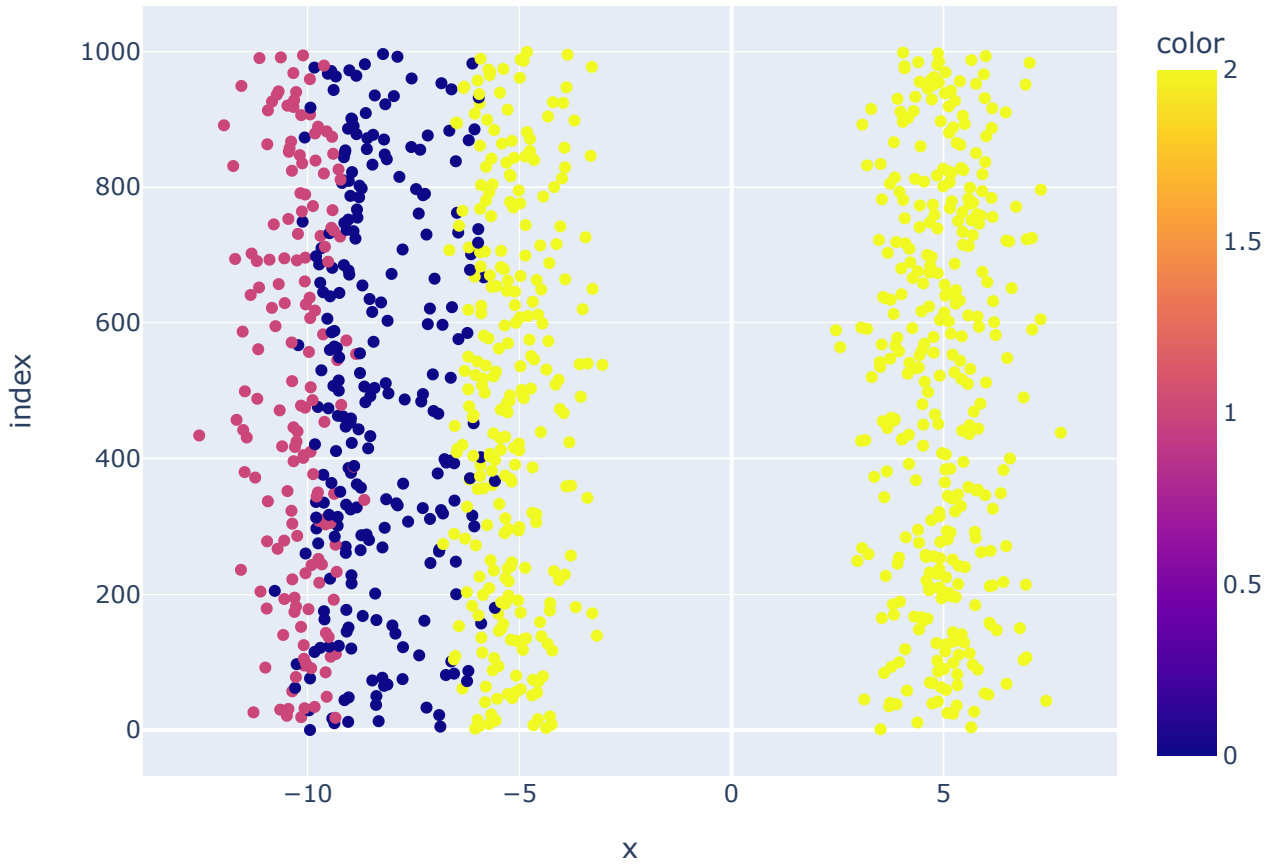
In [6]:
```
class KMeansClustering:
    def __init__(self, X, num_clusters):
        # Initialization of parameters
        self.K = num_clusters # Number of clusters
        self.max_iterations = 100 # Maximum number of iterations to avoid running infini
        self.num_examples, self.num_features = X.shape # num of examples, num of feature
        self.plot_figure = True # Whether to plot figures during the iterations
        self.final_centroids = None  # Variable to store final centroids


    # randomly initialize centroids
    def initialize_random_centroids(self, X):
        centroids = np.zeros((self.K, self.num_features)) # initialize centroids with ze
        for k in range(self.K):
            centroid = X[np.random.choice(range(self.num_examples))] # random centroids
            centroids[k] = centroid
```

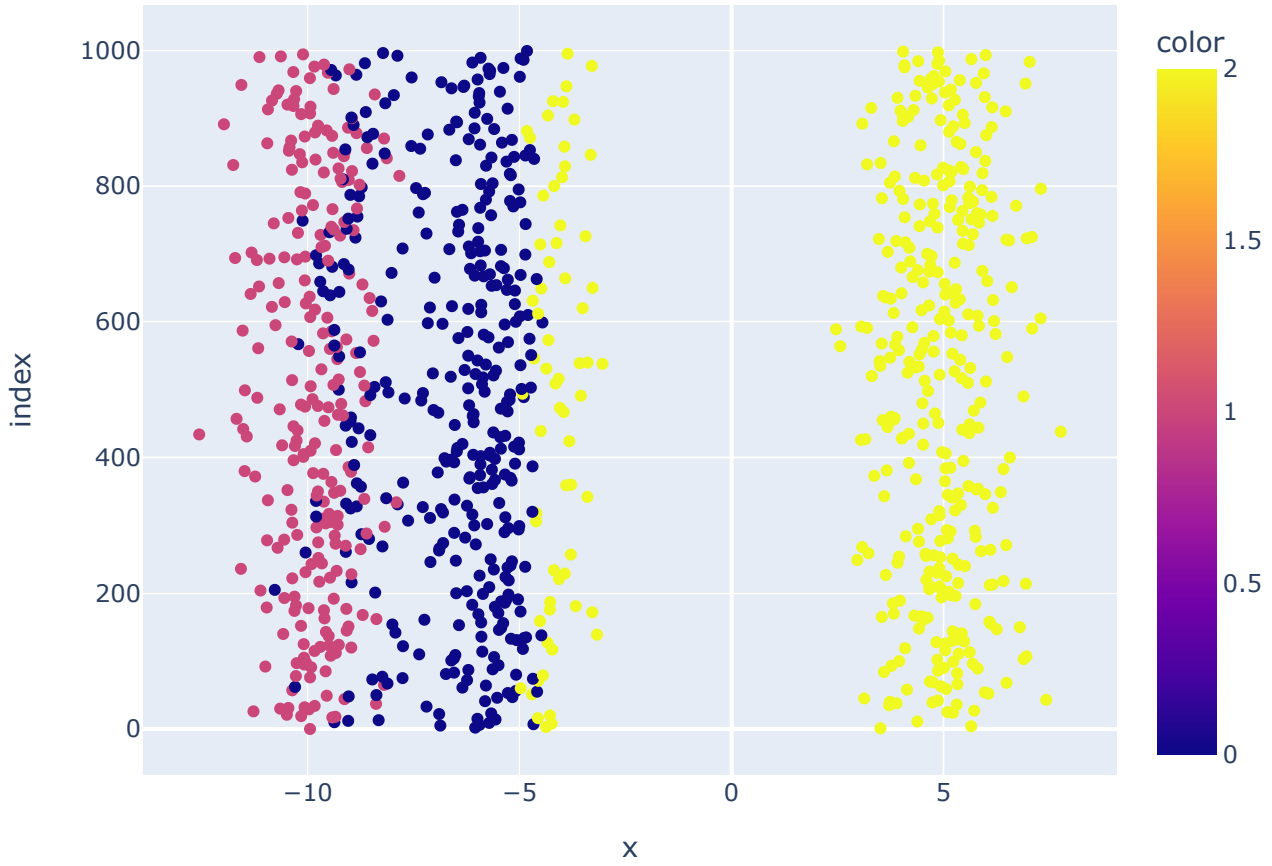
```
        return centroids # return randomly initialized centroids


    # create cluster Function
    def create_cluster(self, X, centroids):
        clusters = [[] for _ in range(self.K)] #initialize clusters
        for point_idx, point in enumerate(X):
            closest_centroid = np.argmin(
            np.sqrt(np.sum((point-centroids)**2,axis=1))
            )
            # Find the closest centroid using Euclidean distance(calculate distance of e
            clusters[closest_centroid].append(point_idx)
        return clusters
    #Calclulate new centroids
    def calculate_new_centroids(self, cluster, X):
        centroids = np.zeros((self.K, self.num_features)) # row , column full with zero
        for idx, cluster in enumerate(cluster):
            new_centroid = np.mean(X[cluster], axis=0) # Calculate the mean for new cent
            centroids[idx] = new_centroid
        return centroids

    # prediction
    def predict_cluster(self, clusters, X):
        y_pred = np.zeros(self.num_examples) # Initialize new centroids with zeros
        for cluster_idx, cluster in enumerate(clusters):
            for sample_idx in cluster:
                y_pred[sample_idx] = cluster_idx
        return y_pred

    # plotinng scatter plot
    def plot_fig(self, X, y):
            fig = px.scatter(X[:, 0], X[:, 1], color=y)
            fig.show() # Visualize the scatter plot
    #fit data
    def fit(self,X):
        centroids = self.initialize_random_centroids(X) # initialize random centroids
        for _ in range(self.max_iterations):
            clusters = self.create_cluster(X, centroids) # create cluster
            previous_centroids = centroids
            centroids = self.calculate_new_centroids(clusters, X) # Calculate new centro
            diff = centroids - previous_centroids # calculate difference
            if not diff.any():
                break   # Break the loop if centroids do not change
            y_pred = self.predict_cluster(clusters, X) # Make predictions
            if self.plot_figure: # If True, plot the scatter plot
                self.plot_fig(X, y_pred)
        self.final_centroids = centroids
        return y_pred
```

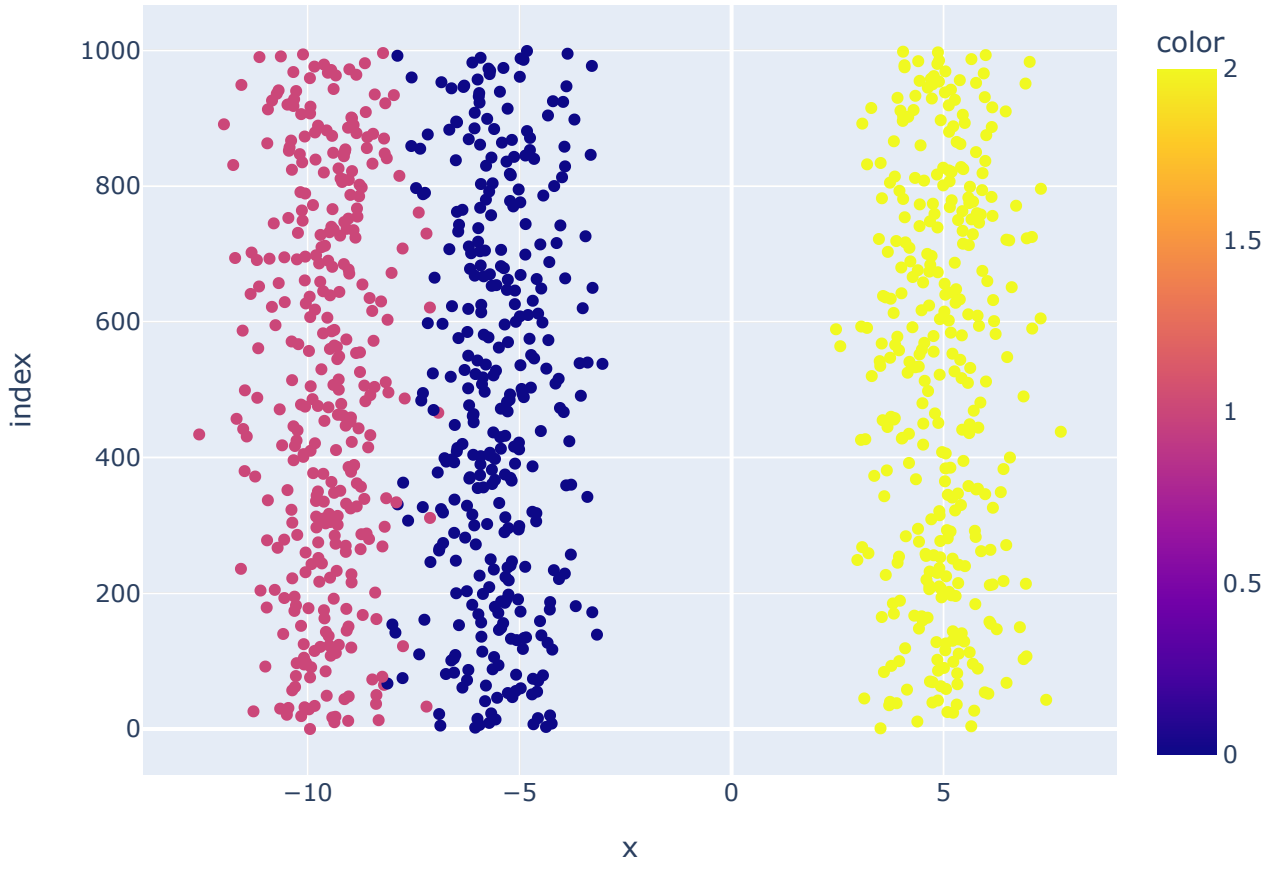
In [7]:

```
if __name__ == "__main__":
    np.random.seed(10) #  Set seed for reproducibility
    num_clusters = 3  # num of cluster
    df = pd.read_csv("HouseholdWealth.csv")

    # Generate synthetic data with make_blobs
    from sklearn.datasets import make_blobs
    X, _ = make_blobs(n_samples=1000, n_features=2, centers=num_clusters, random_state=1
        # Initialize and fit KMeans clustering
    Kmeans = KMeansClustering(X, num_clusters)
    y_pred = Kmeans.fit(X)
#Apparently the figure is not showing in the downloaded pdf or html file, hence use the
```


index
1000
800
600
400
200
0
−10
−5
0
5
x
color
2
1.5
1
0.5
0


index
1000
800
600
400
200
0
−10
−5
0
5
x
color
2
1.5
1
0.5
0


index
1000
800
600
400
200
0
−10
−5
0
5
x
color
2
1.5
1
0.5
0


index
1000
800
600
400
200
0
−10
−5
0
5
x
color
2
1.5
1
0.5
0

In [8]:
```
final_centroids = Kmeans.final_centroids  # Retrieve the final centroids
print("Final Centroids:")
print(final_centroids)  # Print or use these centroids for reporting
```

```
Final Centroids:
[[-0.08676719 -5.53228932]
 [ 5.48944535 -9.58743578]
 [ 2.6320836   4.96394712]]
```

In [9]:
```
from sklearn import preprocessing
from sklearn import cluster
scaler = preprocessing.StandardScaler()
data_scaled = scaler.fit_transform(df)

pd.DataFrame(data_scaled).describe()

kmeans = cluster.KMeans(n_clusters=2, init='k-means++') # 'k-means++' : selects initial
kmeans.fit(data_scaled)
kmeans.inertia_

SSE = []
for i in range(1,20):
    kmeans = cluster.KMeans(n_clusters = i, init='k-means++') # iterate from range (1, 2
    kmeans.fit(data_scaled)
    SSE.append(kmeans.inertia_)

# converting the results into a dataframe and plotting them
frame = pd.DataFrame({'Cluster':range(1,20), 'SSE':SSE})
plt.figure(figsize=(12,6))
plt.plot(frame['Cluster'], frame['SSE'], marker="*")
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.show()

kmeans = cluster.KMeans(n_clusters=5, init='k-means++')
kmeans.fit(data_scaled)
pred = kmeans.predict(data_scaled)
pred
```

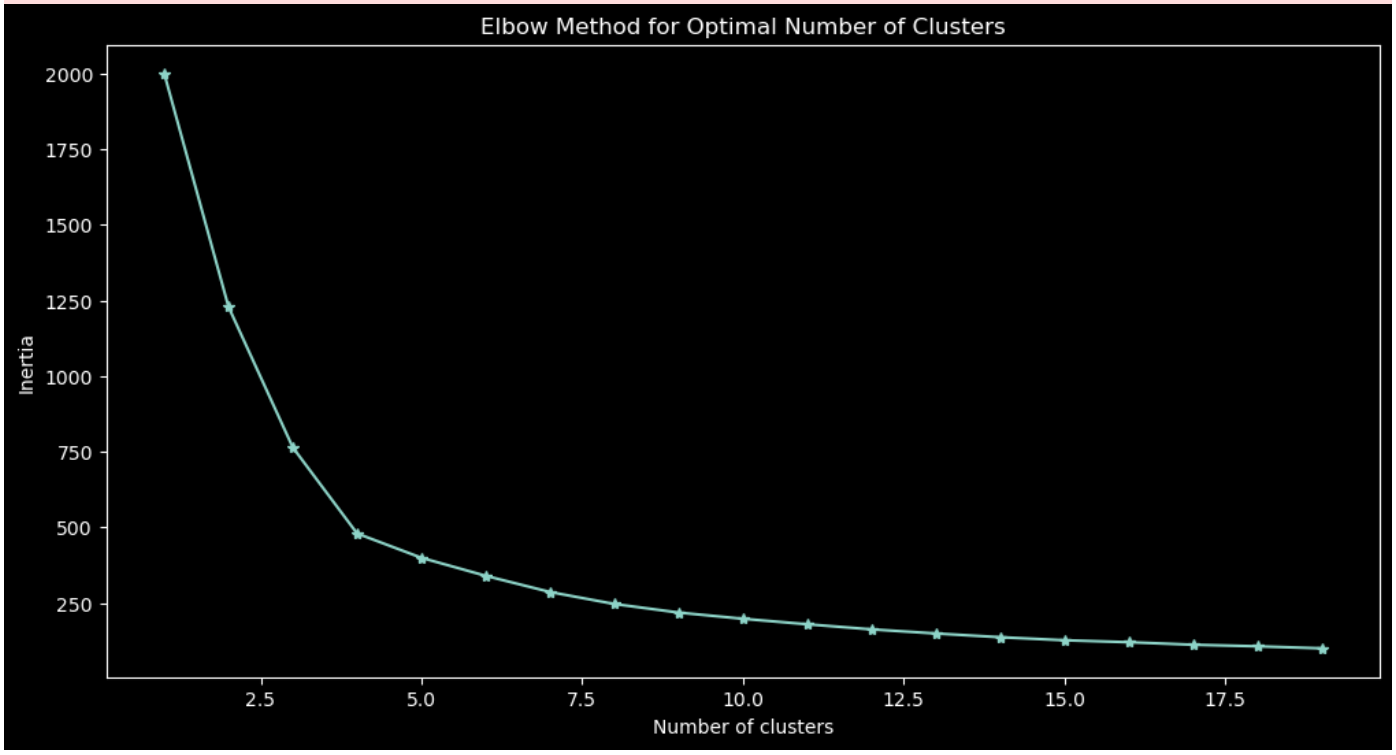
```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:
```

```
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:
```

```
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:
```

```
The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:
```

```
KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.
```




```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarnin
g:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_
init` explicitly to suppress the warning

C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks th
an available threads. You can avoid it by setting the environment variable OMP_NUM_THREA
DS=4.
```

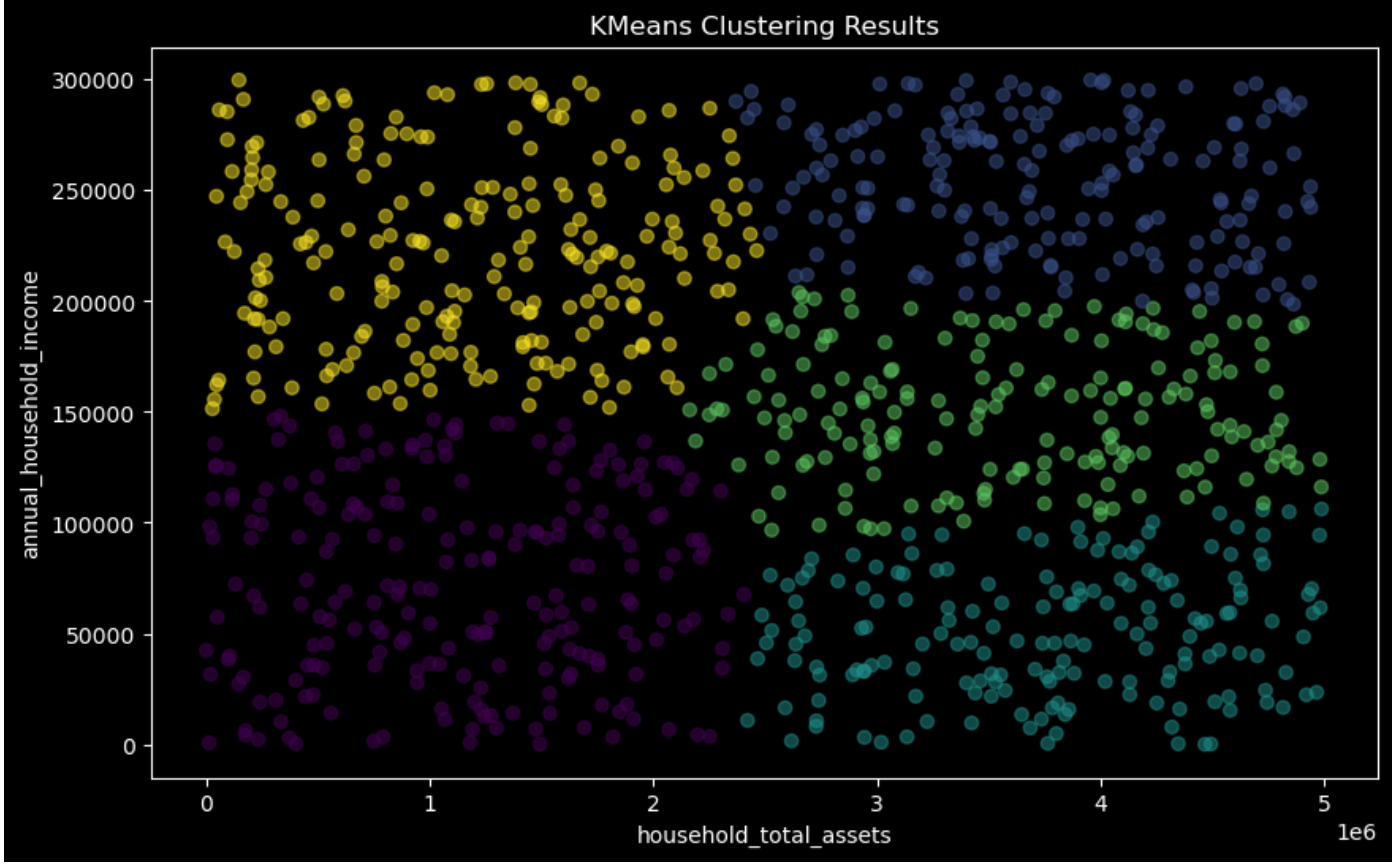
Out[9]:

```
array([0, 3, 3, 4, 0, 1, 1, 4, 2, 1, 3, 1, 0, 4, 2, 0, 4, 4, 4, 2, 3, 0,
       0, 2, 1, 0, 0, 0, 0, 1, 3, 4, 0, 3, 2, 2, 3, 4, 0, 4, 0, 4, 1, 0,
       4, 4, 2, 3, 4, 4, 4, 2, 1, 1, 0, 2, 3, 3, 0, 3, 3, 0, 1, 2, 4, 4,
       2, 3, 4, 0, 0, 4, 0, 3, 3, 1, 1, 3, 1, 1, 2, 1, 0, 4, 0, 3, 4, 1,
       1, 2, 3, 3, 4, 3, 3, 1, 0, 1, 4, 1, 1, 4, 2, 4, 2, 0, 4, 2, 3, 0,
       3, 1, 1, 4, 3, 0, 3, 0, 1, 3, 0, 4, 2, 0, 0, 0, 3, 4, 3, 0, 0, 0,
       4, 4, 0, 0, 1, 3, 3, 3, 1, 4, 4, 4, 0, 1, 2, 1, 0, 0, 1, 4, 3, 0,
       2, 4, 4, 4, 0, 2, 3, 0, 0, 4, 4, 4, 3, 0, 3, 1, 0, 1, 2, 2, 4, 0,
       4, 3, 3, 3, 4, 0, 1, 4, 3, 3, 2, 0, 1, 3, 3, 4, 3, 4, 2, 3, 4, 1,
       1, 4, 1, 2, 4, 1, 3, 3, 4, 4, 2, 1, 3, 0, 3, 4, 3, 0, 4, 4, 0, 0,
       0, 1, 2, 4, 4, 2, 4, 4, 1, 4, 3, 2, 3, 1, 0, 2, 0, 2, 2, 2, 1, 4,
       3, 4, 1, 3, 0, 2, 1, 0, 2, 0, 1, 1, 3, 1, 2, 4, 2, 2, 0, 4, 2, 4,
       0, 1, 3, 1, 0, 0, 3, 1, 1, 0, 3, 4, 1, 0, 0, 4, 2, 3, 4, 0, 1, 2,
       0, 1, 1, 3, 0, 4, 2, 4, 0, 2, 0, 2, 0, 4, 2, 4, 1, 0, 0, 1, 4, 2,
```

```
2, 3, 3, 0, 0, 1, 1, 0, 4, 0, 0, 4, 3, 2, 2, 1, 1, 0, 0, 2, 3, 0,
0, 2, 1, 2, 4, 4, 3, 2, 1, 3, 3, 3, 2, 0, 2, 3, 1, 2, 1, 1, 4, 1,
3, 1, 4, 2, 1, 3, 4, 2, 1, 2, 1, 2, 4, 1, 2, 4, 2, 3, 3, 4, 3, 0,
0, 4, 1, 0, 0, 4, 4, 4, 3, 1, 0, 1, 1, 1, 0, 0, 4, 1, 3, 1, 1, 0,
0, 0, 3, 0, 4, 4, 4, 1, 3, 3, 0, 4, 1, 3, 4, 1, 0, 0, 2, 0, 1, 4,
3, 3, 3, 3, 4, 1, 2, 1, 4, 0, 4, 0, 3, 3, 4, 0, 4, 1, 4, 0, 1, 3,
4, 2, 2, 1, 3, 4, 1, 0, 3, 2, 2, 1, 0, 0, 2, 0, 4, 3, 3, 1, 2, 3,
4, 4, 1, 1, 2, 0, 2, 1, 1, 4, 2, 1, 4, 4, 3, 3, 0, 3, 1, 3, 4, 4,
3, 2, 4, 4, 3, 3, 1, 0, 1, 0, 2, 4, 0, 4, 4, 3, 0, 1, 4, 2, 2, 0,
2, 3, 4, 2, 2, 0, 0, 0, 0, 3, 4, 2, 1, 0, 4, 4, 2, 0, 3, 2, 1, 1,
3, 2, 3, 3, 0, 3, 0, 0, 1, 4, 2, 4, 0, 2, 0, 3, 1, 3, 2, 2, 1, 1,
2, 4, 1, 3, 1, 1, 3, 4, 2, 4, 2, 3, 4, 3, 2, 0, 3, 4, 2, 2, 1, 3,
1, 4, 1, 4, 3, 0, 4, 0, 0, 3, 2, 1, 0, 4, 2, 4, 2, 0, 4, 4, 4, 2,
3, 0, 3, 0, 4, 0, 0, 3, 0, 4, 2, 0, 4, 1, 0, 0, 3, 2, 1, 2, 4, 3,
3, 1, 4, 3, 4, 0, 2, 0, 3, 0, 0, 4, 0, 0, 1, 4, 1, 2, 4, 1, 1, 2,
2, 1, 3, 4, 2, 3, 2, 4, 3, 0, 0, 2, 0, 0, 0, 2, 0, 4, 0, 4, 2, 2,
4, 2, 1, 4, 1, 3, 2, 4, 3, 2, 0, 3, 3, 1, 0, 3, 1, 1, 1, 0, 4, 3,
3, 1, 1, 3, 1, 1, 3, 2, 2, 2, 4, 2, 0, 3, 4, 2, 1, 4, 0, 2, 1, 3,
3, 4, 4, 1, 4, 4, 4, 2, 1, 4, 3, 2, 1, 0, 0, 4, 4, 0, 4, 4, 1, 0,
1, 2, 4, 1, 0, 3, 3, 0, 1, 0, 4, 4, 4, 3, 0, 1, 1, 0, 3, 0, 0, 1,
4, 1, 1, 0, 2, 1, 1, 0, 4, 3, 3, 2, 0, 1, 2, 4, 0, 0, 4, 2, 1, 2,
2, 1, 1, 1, 0, 4, 4, 0, 0, 2, 4, 4, 2, 3, 3, 4, 3, 4, 0, 0, 2, 0,
2, 1, 0, 0, 2, 3, 2, 2, 1, 1, 0, 4, 0, 1, 0, 0, 1, 4, 0, 2, 2, 0,
4, 2, 0, 0, 4, 4, 2, 0, 0, 4, 3, 2, 0, 4, 3, 0, 2, 4, 1, 2, 4, 2,
4, 1, 2, 4, 4, 3, 0, 0, 1, 0, 4, 3, 4, 1, 3, 2, 0, 2, 2, 3, 1, 2,
2, 2, 2, 4, 4, 4, 3, 1, 3, 2, 4, 0, 0, 4, 1, 0, 0, 2, 0, 1, 3, 4,
2, 3, 0, 3, 0, 1, 2, 4, 3, 3, 2, 0, 4, 4, 3, 3, 4, 4, 4, 4, 3, 3,
2, 1, 1, 4, 1, 1, 0, 1, 3, 3, 0, 4, 0, 3, 1, 1, 1, 1, 4, 4, 2, 0,
2, 1, 4, 3, 3, 2, 1, 0, 0, 1, 0, 1, 3, 3, 1, 1, 0, 1, 1, 1, 3, 2,
4, 0, 4, 4, 2, 2, 1, 1, 4, 0, 4, 1, 4, 0, 4, 2, 2, 2, 4, 0, 3, 0,
4, 2, 3, 1, 2, 2, 3, 1, 0, 1, 0, 4, 2, 4, 4, 3, 3, 0, 0, 4, 3, 4,
2, 3, 0, 3, 3, 1, 2, 4, 0, 0])
```

In [10]:

```
# Visualize the clustered data on the original scale if needed
# For example, you can create a scatter plot with colors representing clusters
plt.figure(figsize=(10, 6))
plt.scatter(df['household_total_assets'], df['annual_household_income'], c=pred, cmap='v
plt.title('KMeans Clustering Results')
plt.xlabel('household_total_assets')
plt.ylabel('annual_household_income')
plt.show()
```


KMeans Clustering Results
annual_household_income
300000
250000
200000
150000
100000
50000
0
0
1
2
3
4
5
household_total_assets
1e6