2_ClusterAnalysis

August 27, 2024

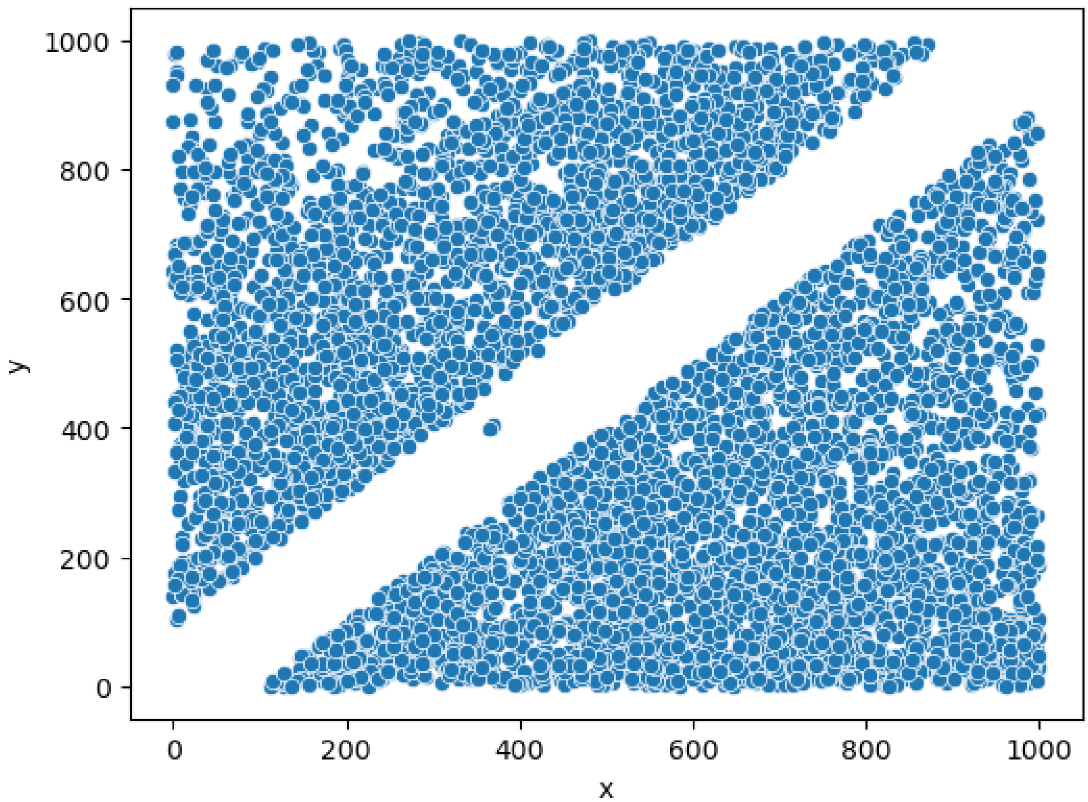
[6]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn import cluster
from sklearn.metrics import pairwise_distances_argmin

# The objective is to identify the optimal locations for a series of drone␣
↪depots, using the coordinates of the clients as a reference

# Assigning the number of created clusters
NUM_OF_CLUSTERS = 3

# Reading the file
filePath = "drone_delivery_v1.csv"
data = pd.read_csv(filePath, sep=";")

# Printing the basic statistics
data.describe()
```

[6]:

	clientid	x	y
count	5956.000000	5956.000000	5956.000000
mean	2978.500000	508.823177	427.554772
std	1719.493433	271.061462	289.044640
min	1.000000	0.017692	0.043285
25%	1489.750000	282.582920	170.079921
50%	2978.500000	518.100892	397.786441
75%	4467.250000	727.156497	669.982518
max	5956.000000	999.533215	999.731720

[8]:
```python
# Create a scatter plot using seaborn
sns.scatterplot(data=data, x='x', y='y')
```

[8]: <Axes: xlabel='x', ylabel='y'>

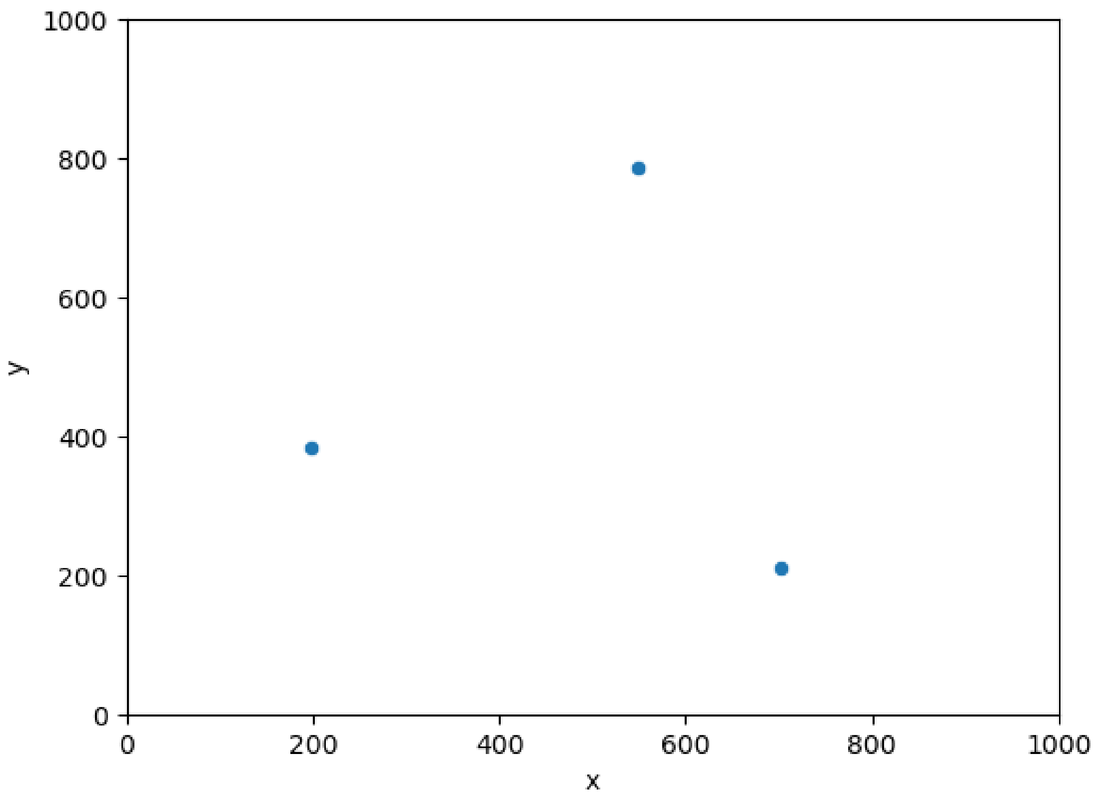


[28]:
```
# Creating K-Means model and fitting it to the data
kmeans = cluster.KMeans(n_clusters=NUM_OF_CLUSTERS, n_init=10)
kmeans.fit(data[['x', 'y']])

# Finding cluster centroids
centroids = kmeans.cluster_centers_
print(centroids)
```

```
[[198.54073778 383.08097795]
 [702.21311616 211.32734145]
 [548.20586479 787.2788963 ]]
```

[44]:
```
# Creating a dataframe from the centroids and visualising the locations of the
↪centroids with a scatterplot
centroid_df = pd.DataFrame(centroids, columns=["x","y"])
values = sns.scatterplot(data=centroid_df, x='x', y='y')
values.set_xlim(0, 1000)  # Set x-axis limit
values.set_ylim(0, 1000)  # Set y-axis limit
```

[44]: (0.0, 1000.0)



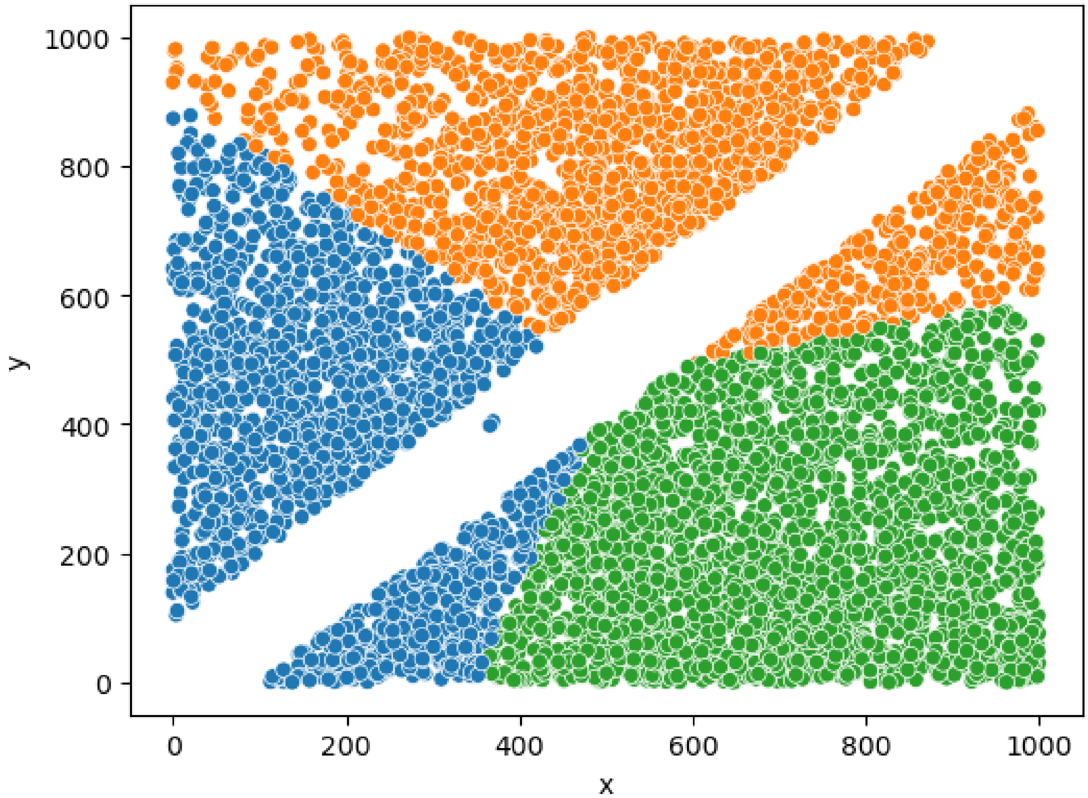
[24]:
```
# Compute the index of the nearest centroid for each data point and print the
↪head of the data
nearest_centroid = pairwise_distances_argmin(data[['x', 'y']], centroids)
data['cluster'] = nearest_centroid
print(data.head(10))
```

```
   clientid           x           y  cluster
0         1  622.771572  164.857623        2
1         2  416.357298  630.193634        1
2         3  292.735020  567.333231        0
3         4  737.211288  166.225676        2
4         5  540.475375  682.912298        1
5         6  535.469492  318.439661        2
6         7  640.380050  870.833221        1
7         8  235.772075  359.048203        0
8         9  481.896884  661.491838        1
9        10  730.032789  312.177817        2
```

[34]:
```
# Creating a scatterplot from all the data values and visualising the clusters
↪using different colours
data['cluster'] = pd.Categorical(data['cluster'])
```

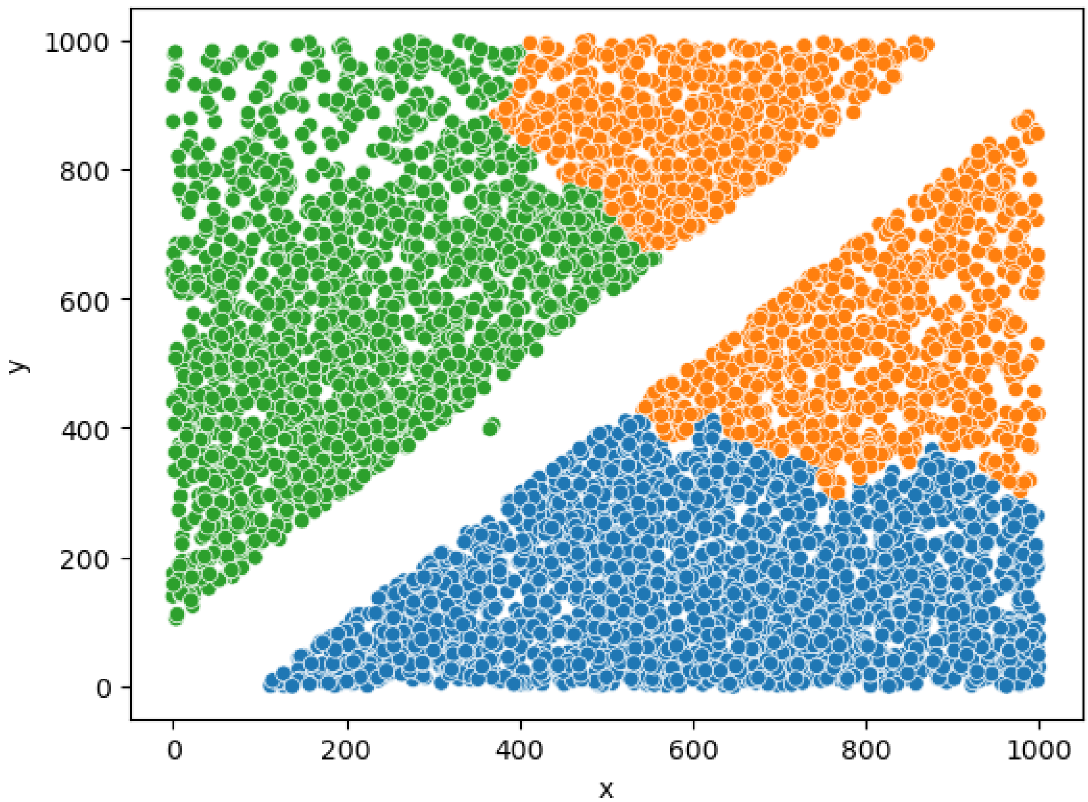
```
temp = sns.scatterplot(data=data, x='x', y='y', hue='cluster')
temp.legend([], [], frameon=False)
```

[34]: <matplotlib.legend.Legend at 0x1f45d34af90>



[38]:
```
# Creating a scatterplot using Agglomerative hierarchical clustering instead of␣
↪K-Means
agglomerative = cluster.AgglomerativeClustering(n_clusters=NUM_OF_CLUSTERS)
data['cluster'] = agglomerative.fit_predict(data[['x', 'y']])

# Creating a scatterplot from all the data values and visualising the clusters␣
↪using different colours
data['cluster'] = pd.Categorical(data['cluster'])
temp = sns.scatterplot(data=data, x='x', y='y', hue='cluster')
temp.legend([], [], frameon=False)
```

[38]: <matplotlib.legend.Legend at 0x1f45d39bb90>



[46]:
```
# When comparing K-Means and Agglomerative clustering, there is a notable
↪difference in the results.
# We liked the way agglomerative clustering divided the regions in a more
↪sensible way, at least with 3 clusters.
```