# DHW-Exercise_E10.3-k-Means-Clustering_Example

April 21, 2021

## 1 Step1: Introduction to DWH-Exercise E10.3 k-Means Clustering

Exercise E10.3 from Exercises to Lesson “DW10-Advanced Analysis II” of the lecture “Introduction to Data Warehousing” at DHBW Stuttgart (SS2021.)

**by Luis Seybold and Sven Stahl (DHBW Stuttgart); 19. April 2021** Reviewed and extended by Dr. Hermann Völlinger (DHBW Stuttgart); 21. April 2021

The k-means algorithm of the libraray sklearn.cluster is used to generate k-means clusters for the 19 datapoints of the exercise text. We want 3 clusters, so we set k=3 (= number of centroids). We will import these datapoints (dataset) in Step2 with a 2-dimensional dataframe.

**The program is structured in 6 steps:**

1. Introduction of the Exercise E10.3
2. Import the packages and classes you need.
3. Provide data to work with and eventually do appropriate transformations
4. Define the Visualization of the data clusters
5. Execute the K-means clustering algorithmus
6. Summary and final remarks

Prerequisites: input data - given in the program; images - all images are located in the directory ‘Images/’

[1]:
```python
# Indrodruction - Print the execution plan
print(" We start the execution of the 6 steps:")
print(" 1.        Introduction of the Exercise E10.3")
print(" 2.        Import the packages and classes you need")
print(" 3.        Provide data to work with and eventually do appropriate␣
 ↪transformations")
print(" 4.        Define the visualization of the data clusters")
print(" 5.        Execute the K-means clustering algorithmus")
print(" 6.        Summary and final remarks")
```

```
 We start the execution of the 6 steps:
 1.     Introduction of the Exercise E10.3
 2.     Import the packages and classes you need
 3.     Provide data to work with and eventually do appropriate transformations
```

```
4.     Define the visualization of the data clusters
5.     Execute the K-means clustering algorithmus
6.     Summary and final remarks
```

## 2 Step2: Import the needed libraries

sklearn - sklearn for the algorithm implementation

pandas - loads the dataset and provids necessary frame details. pandas is also uese for the dataset management

mathplotlip - matplotlib for plotting the results and steps of the algorithm.

pprint - prints the dictionary storage

sys - version information to pythonImport of libraries

[2]:
```python
# libraries to import
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import sklearn as sk
import pandas as pd
import matplotlib.pyplot as plt
import pprint

# python version check library
import sys

# to check the time of execution, import function time
import time
import datetime

print(f"This notebook is now launched: {datetime.datetime.now()}")
print()
print("Versions of the used runtime and libraries:")

# print python version, for some imports this version number is viewed as␣
 ↪theirs.
print("python {}".format(sys.version))

# print sklearn
print("sklearn {}".format(sk.__version__))

# print pandas version
print("pandas {}".format(pd.__version__))
```

```
This notebook is now launched: 2021-04-21 18:51:57.970680

Versions of the used runtime and libraries:
```

```
python 3.7.6 (default, Jan  8 2020, 20:23:39) [MSC v.1916 64 bit (AMD64)]
sklearn 0.22.1
pandas 1.0.1
```

## 3 Step3: Import the data

The data (i.e. 19 datapoints) is given in the program in a 2-dimensional dataframe.

[3]:
```
# Dataset declaration
df = pd.DataFrame({
    'x': [12, 20, 28, 18, 29, 33, 24, 45, 45, 52, 51, 52, 55, 53, 55, 61, 64,␣
↪69, 72],
    'y': [39, 36, 30, 52, 54, 46, 55, 59, 63, 70, 66, 63, 58, 23, 14, 8, 19, 7,␣
↪24]
})
```

## 4 Step4: Define the visualization of the k-means clusters

This function is used to plot the output of the k-means algorithm. It plots three clusters with it centers. For this it takes the clusters, the k-means-model and the current iteration that is to be plotted. Dependant to the expected outcome we define the size/shape of the plot. #### Remark: If we use another number k (i.e. k greater than 4) the size and shape of the plot shoud be changed , s.t. all centroids are visible.

[4]:
```
# plotting function
def plot_clusters(clusters, clf, iteration):
    fig, ax = plt.subplots(figsize=(15,7))
    plt.scatter(x=clusters[clusters['Cluster_Prediction'] == 0]['x'],
                y=clusters[clusters['Cluster_Prediction'] == 0]['y'],
                s=70,edgecolor='teal', linewidth=0.3, c='teal', label='Cluster␣
↪1')


    plt.scatter(x=clusters[clusters['Cluster_Prediction'] == 1]['x'],
                y=clusters[clusters['Cluster_Prediction'] == 1]['y'],
                s=70,edgecolor='lime', linewidth=0.3, c='lime', label='Cluster␣
↪2')


    plt.scatter(x=clusters[clusters['Cluster_Prediction'] == 2]['x'],
                y=clusters[clusters['Cluster_Prediction'] == 2]['y'],
                s=70,edgecolor='magenta', linewidth=0.3, c='magenta',␣
↪label='Cluster 3')
```

```
    plt.scatter(x=clf.cluster_centers_[:, 0], y=clf.cluster_centers_[:, 1], s =␣
↪170, c = 'black', label = 'Centroids',edgecolor='black', linewidth=0.3)
    plt.legend(loc='upper right')
    ax.set_ylabel('Y')
    ax.set_xlabel('X')
    plt.title('Clusters iteration ' + str(iteration), fontsize = 20)
    plt.show()
```

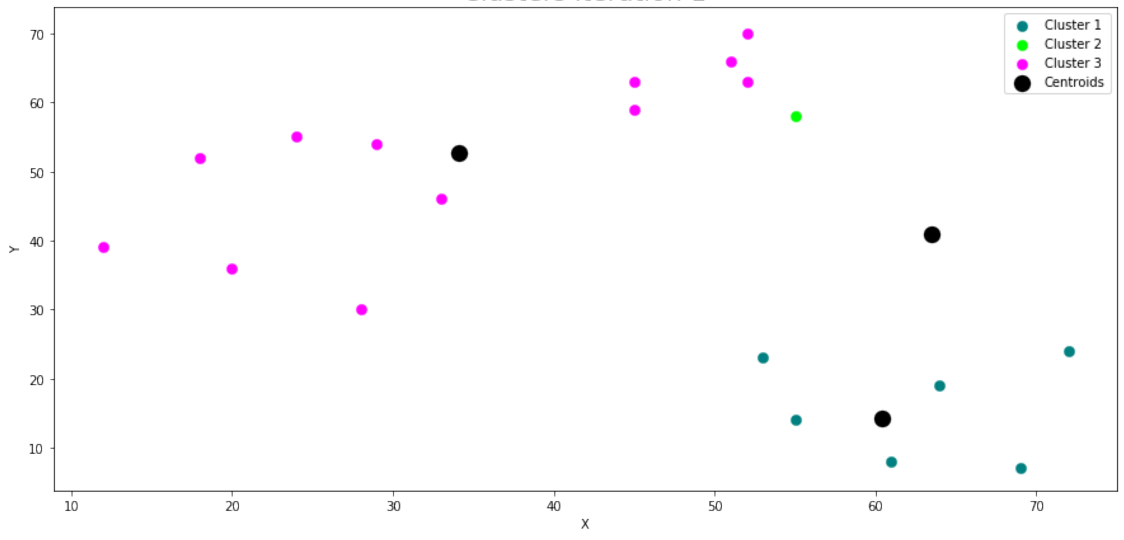
## 5 Step5: Execute the k-means clustering algorithm

In this cell we calculate and plot iterations of the k-means algorithm. For every step we fit our StandardScaler and the k-means. We choose random initializations, three clusters and cap the maximum iterations to the number of steps. As a result the k-means is only calculated up to the given iteration. We plot the output of the k-means with the above function from Step3. The detailed mode for the k-means alogrithm shows us that the algorithm converges at the fourth iteration. After that we have our final centers. The proof of the convergence of this method is outside the scope of this particular task.

[5]:
```
# plot 4 steps of kmeans
for it in range(1, 6):
    scaler = StandardScaler().fit(df)
    kmeans = KMeans(n_clusters=3, init='random', max_iter=it, n_init=1,␣
↪random_state=10, verbose=1)
    clusters = df.copy()
    clusters['Cluster_Prediction'] = kmeans.fit_predict(df)
    plot_clusters(clusters, kmeans, iteration=it)
```

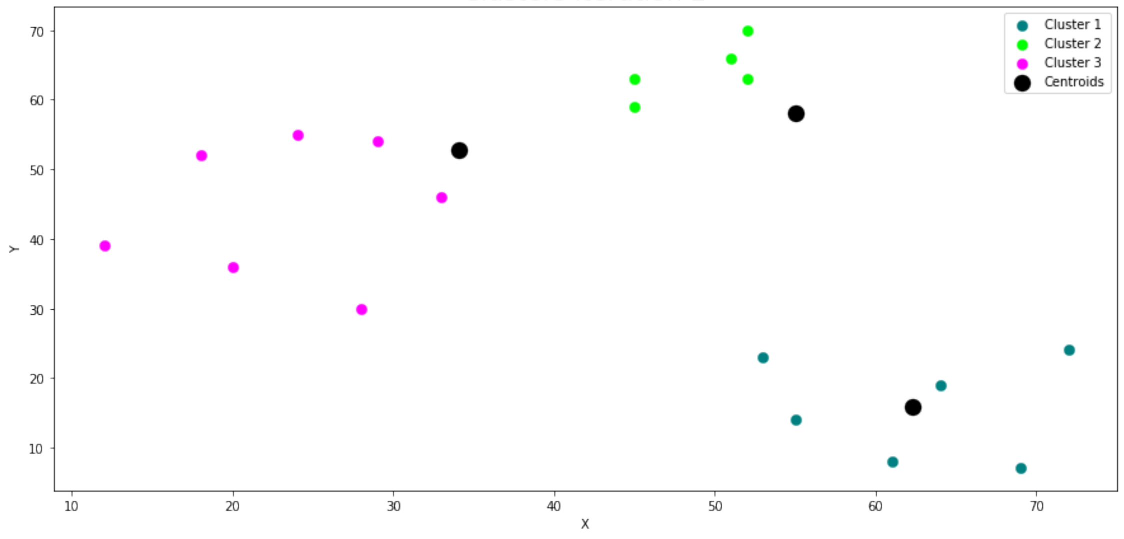
```
Initialization complete
start iteration
done sorting
end inner loop
Iteration 0, inertia 5123.666666666667
```

Clusters iteration 1

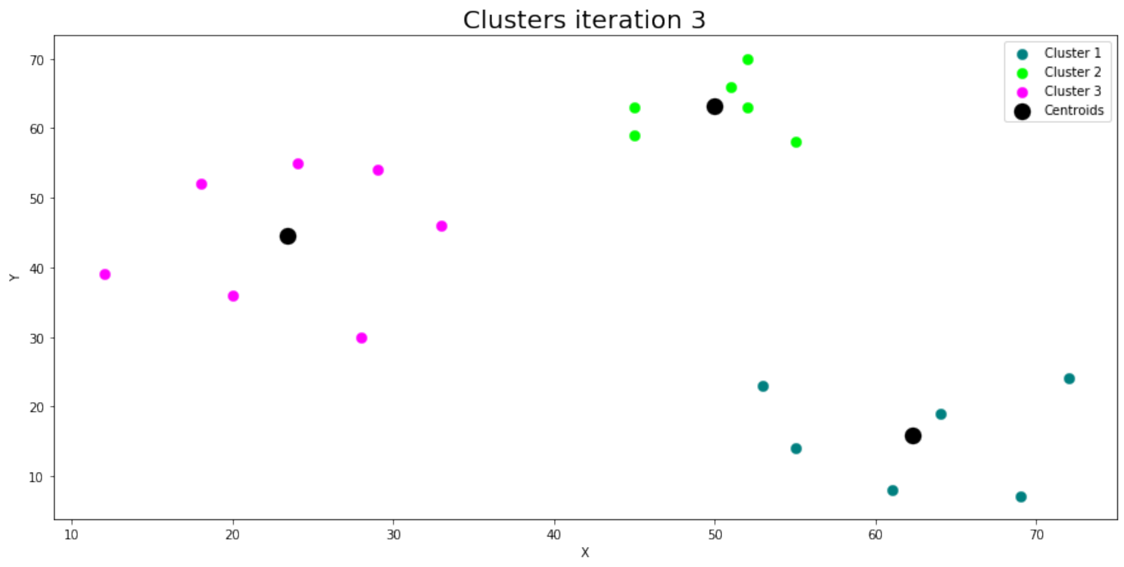


```
Initialization complete
start iteration
done sorting
end inner loop
Iteration 0, inertia 5123.666666666667
start iteration
done sorting
end inner loop
Iteration 1, inertia 4593.333333333331
```

Clusters iteration 2

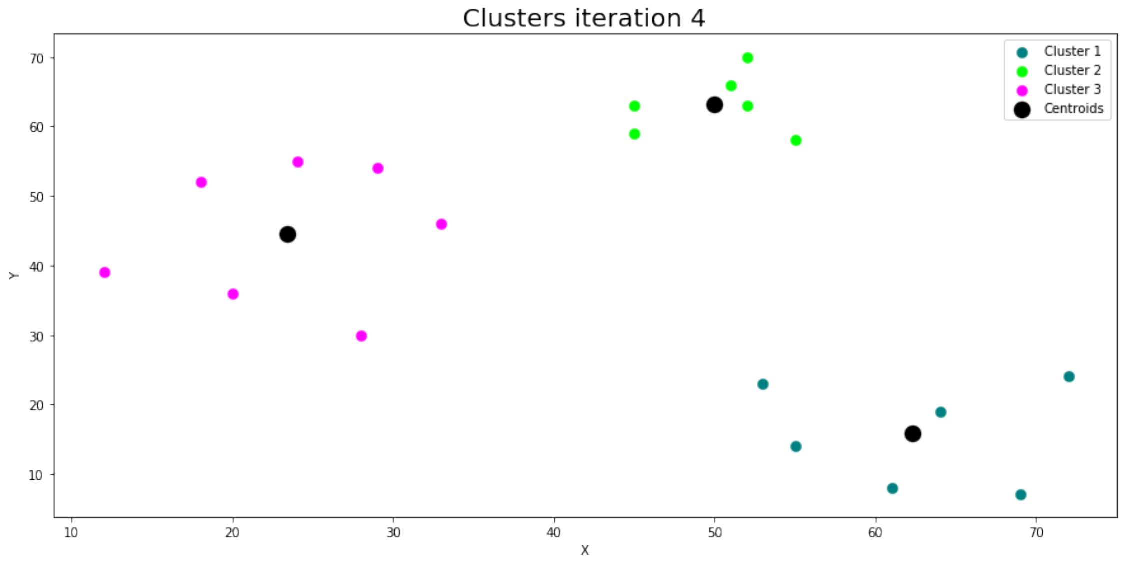


```
Initialization complete
start iteration
done sorting
end inner loop
Iteration 0, inertia 5123.666666666667
start iteration
done sorting
end inner loop
Iteration 1, inertia 4593.333333333331
start iteration
done sorting
end inner loop
Iteration 2, inertia 1624.4285714285716
```

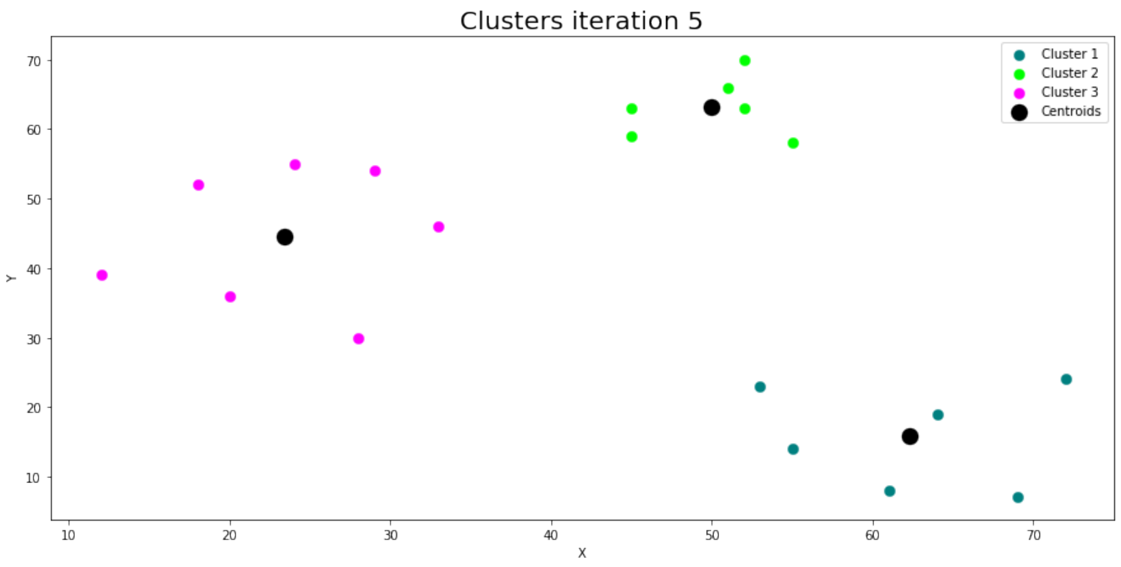


```
Initialization complete
start iteration
done sorting
end inner loop
Iteration 0, inertia 5123.666666666667
start iteration
done sorting
end inner loop
Iteration 1, inertia 4593.333333333331
start iteration
done sorting
end inner loop
Iteration 2, inertia 1624.4285714285716
start iteration
done sorting
```

```
end inner loop
Iteration 3, inertia 1624.4285714285716
center shift 0.000000e+00 within tolerance 3.593213e-02
```



```
Initialization complete
start iteration
done sorting
end inner loop
Iteration 0, inertia 5123.666666666667
start iteration
done sorting
end inner loop
Iteration 1, inertia 4593.333333333331
start iteration
done sorting
end inner loop
Iteration 2, inertia 1624.4285714285716
start iteration
done sorting
end inner loop
Iteration 3, inertia 1624.4285714285716
center shift 0.000000e+00 within tolerance 3.593213e-02
```



## 6 Step6: Summary and final remarks

**Remark1:** We stopped the algorithm by defining that only 5 iterations should run. We know by experience that then the centroids are in a stable location for these set of data. Actually the algorithm is stable after 3 iterations (look on the calculated parameter “center shift”). It is also possible to let the algorithm calculate the numbers of iterations by using this parameter.

**Remark2:** By mathematical methods we should also be able to proof, that the k-means algorithm will convert to stable centroid locations. But this is outside the problem scope of this special exercise.

Remark1 and remark2 will be done in the next version of the program.

Finally we print the execution-time and execution-date.

[6]:
```
# print current date and time
print("date",time.strftime("%d.%m.%Y %H:%M:%S"))
print ("*** End of Homework-E10.3_K-Means Clustering ***")
```

```
date 21.04.2021 18:51:59
*** End of Homework-E10.3_K-Means Clustering ***
```