# Classifiers

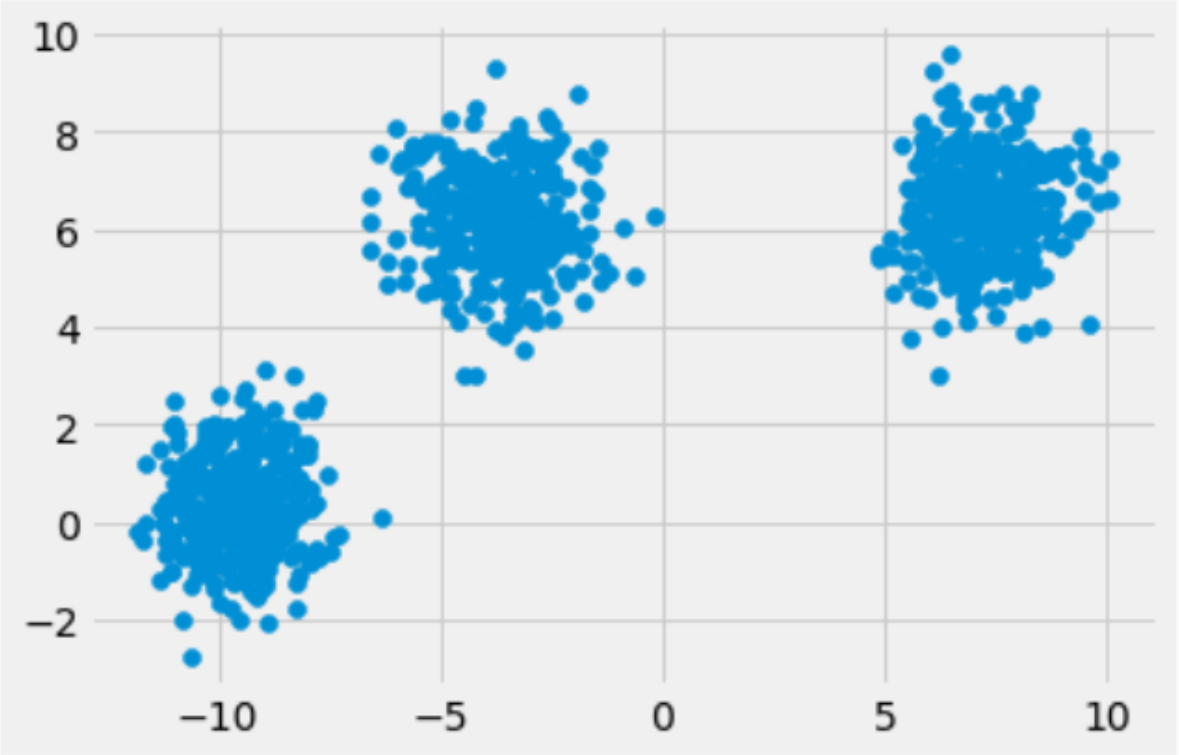
## Report 5

### Data Clustering I


**Author:**
Piotr Pawełko
Piotr Wojsa

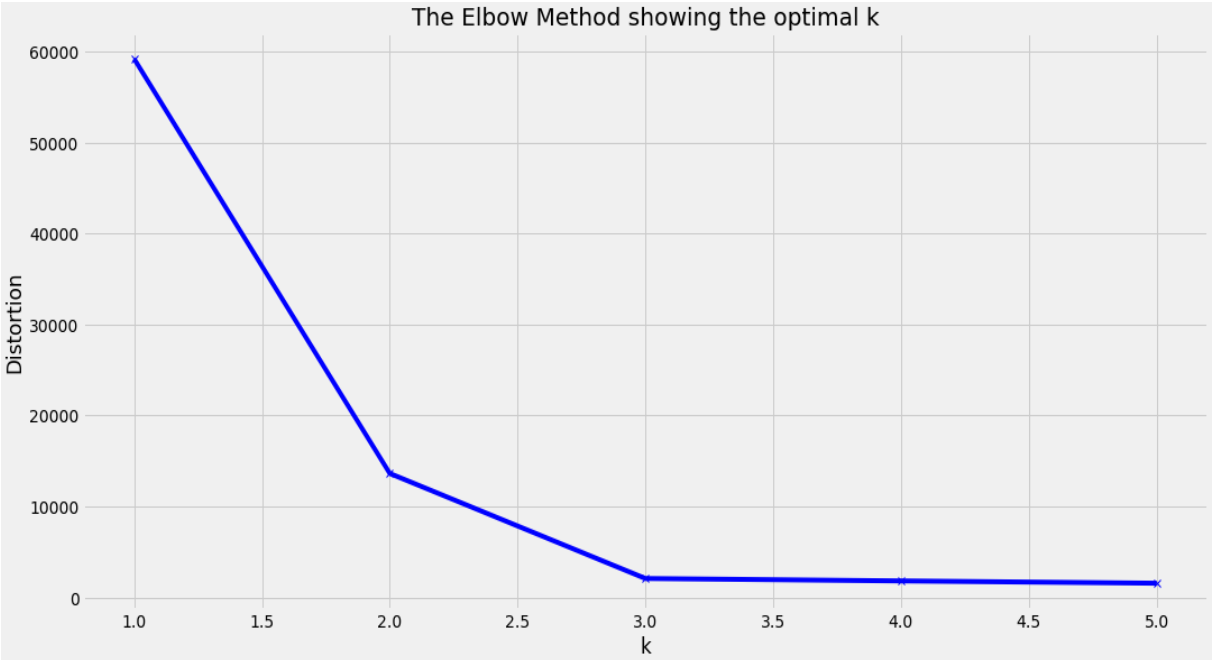
**Laboratory date:**
**27.05.2022**

1. **Generating synthetic dataset containing 1000 samples and 2 features**

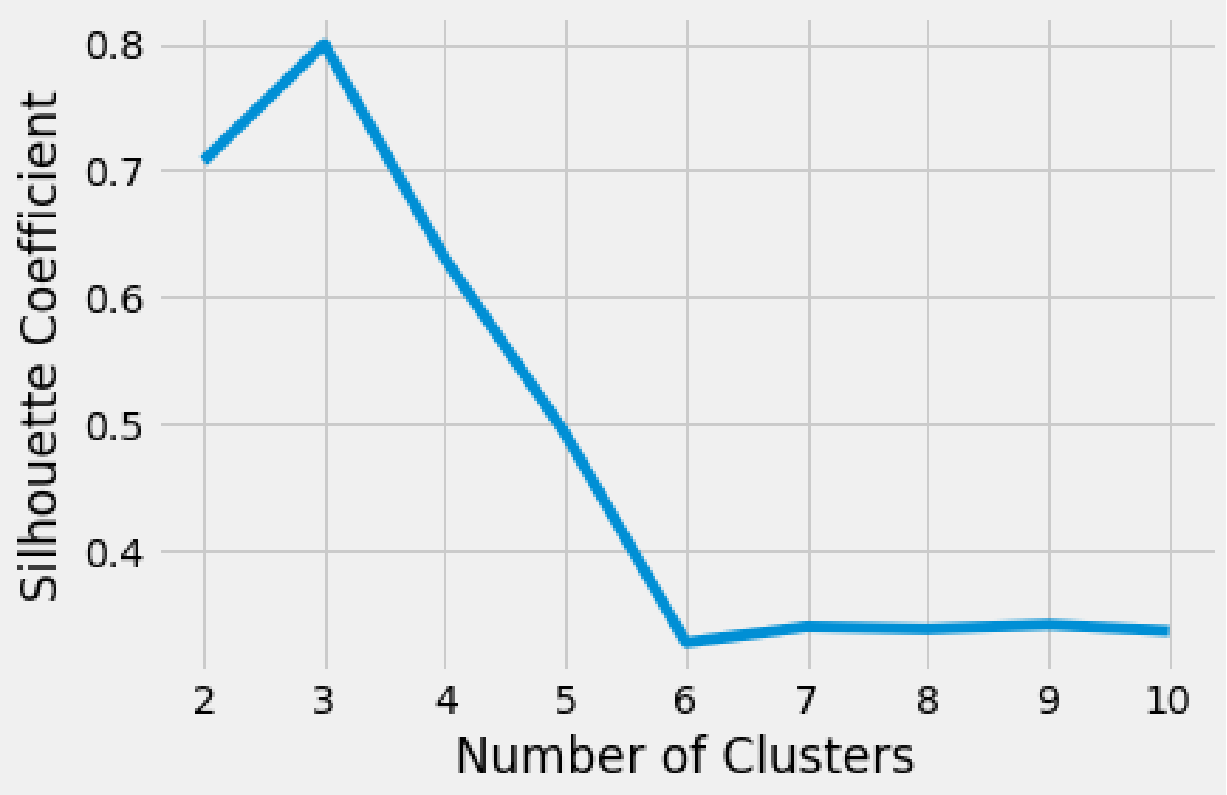



2. **Determining the optimal number of clusters using elbow, silhouette and gap statistic methods**

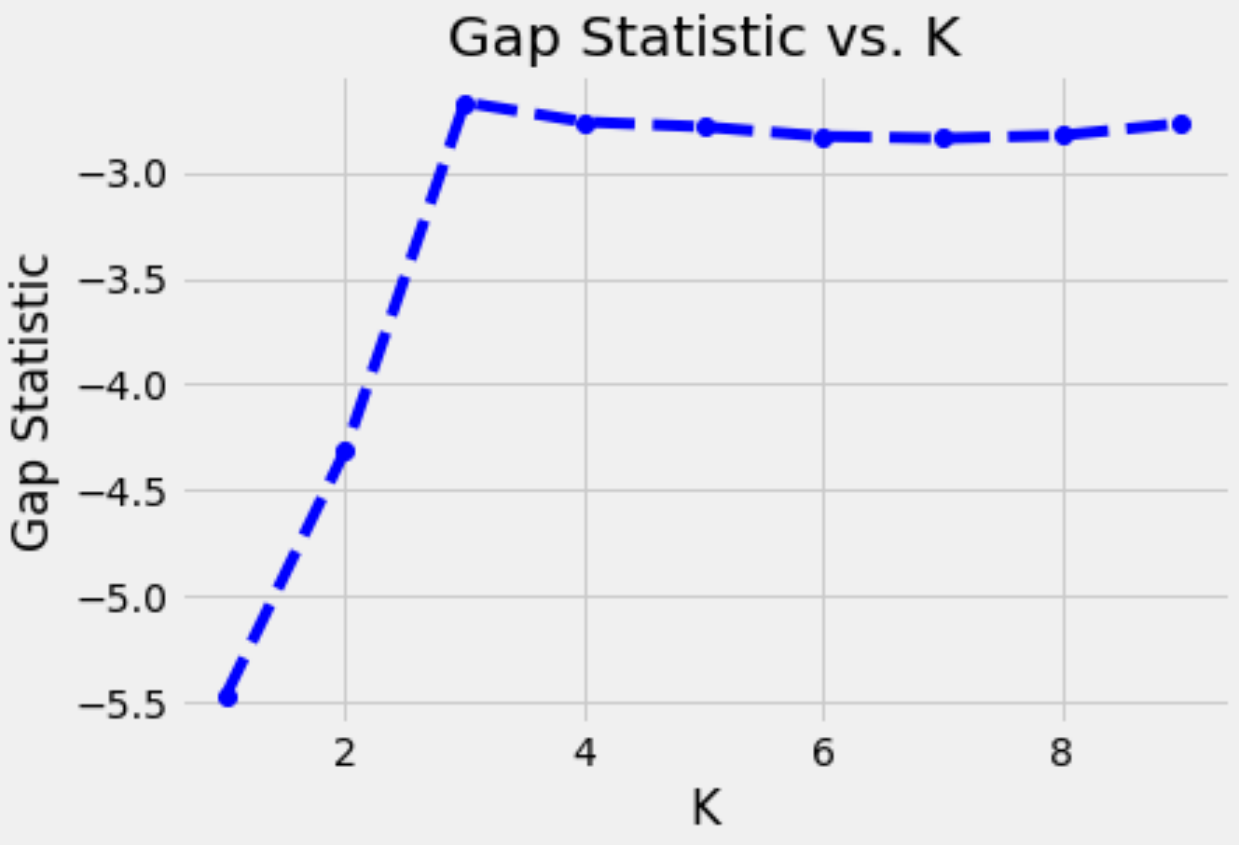
**Elbow method:**



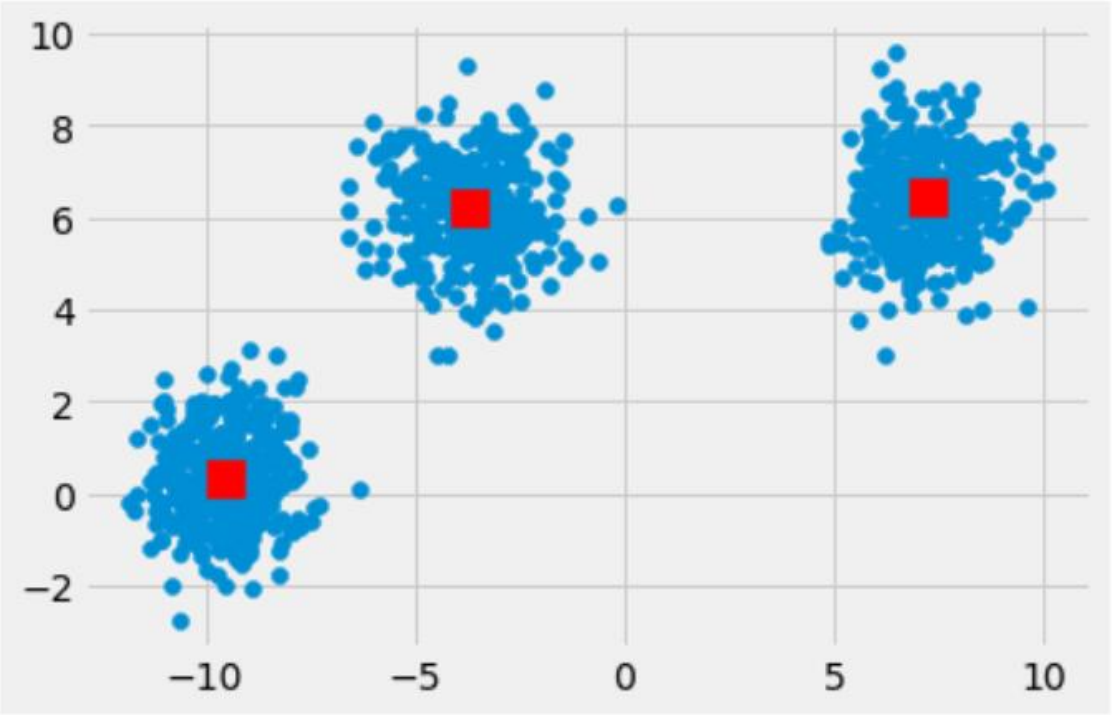
**Silhouette method:**




**Gap statistic method:**



## 3. Performing k-means clustering




## 4. Conclusion

The most popular method for determining number of clusters is elbow method and this is method that we did choose to test first. Elbow method is based on calculating the Within-Cluster-Sum of Squared Errors(WSS) for different number of clusters(k) and selecting the k for which change in WSS first starts to diminish. For generated dataset with 1000 samples, 2 features and 3 centers in our case WSS value started to diminish for k = 3, which is correct. Second method that was tested is silhouette method, which is defined as follows S(i) = [b(i) – a(i)] / max{a(i), b(i)}, where b(i) is the smallest average distance of point I to all points in any other cluster and a(i) is the average distance of i from all other points in its cluster.[1] This method also calculated the number of clusters correctly. Last method that was tested during laboratory is Gap Statistic method, the idea behind this method is to find a way to standardize the comparison of $\log W_k$ with a null reference distribution of the data, i.e. a distribution with no obvious clustering. The estimate for the optimal number of clusters $K$ is the value for which $\log W_k$ falls the farthest below this reference curve. This information is contained in the following formula for the gap statistic: $Gap_n(k) = E_n\{\log W_k\} – \log W_k$.[2] As well as other methods used in this laboratory, gap statistic method determined the number of clusters correctly.

[1] https://towardsdatascience.com/cheat-sheet-to-implementing-7-methods-for-selecting-optimal-number-of-clusters-in-python-898241e1d6ad
[2] https://datasciencelab.wordpress.com/tag/gap-statistic/

## 5. Code

```
# %%
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import make_blobs

# %%
X, y = make_blobs(n_samples=1000, centers=3, n_features=2)
df = pd.DataFrame(dict(x=X[:,0], y=X[:,1], label=y))
plt.scatter(df.x,df.y)

# %%
from sklearn.cluster import KMeans
distortions = []
K = range(1,6)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(df)
    distortions.append(kmeanModel.inertia_)

# %%
plt.figure(figsize=(16,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()

# %%
from sklearn.metrics import silhouette_score

silhouette_coefficients = []

for k in range(2, 11):
   kmeans = KMeans(n_clusters=k)
   kmeans.fit(df)
   score = silhouette_score(df, kmeans.labels_)
   silhouette_coefficients.append(score)
plt.style.use("fivethirtyeight")
plt.plot(range(2, 11), silhouette_coefficients)
plt.xticks(range(2, 11))
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Coefficient")
plt.show()
```

```
# %%
def optimalK(data, nrefs=3, maxClusters=15):
    gaps = np.zeros((len(range(1, maxClusters)),))
    resultsdf = pd.DataFrame({'clusterCount':[], 'gap':[]})
    for gap_index, k in enumerate(range(1, maxClusters)):
# Holder for reference dispersion results
        refDisps = np.zeros(nrefs)
        for i in range(nrefs):

            # Create new random reference set
            randomReference = np.random.random_sample(size=data.shape)

            # Fit to it
            km = KMeans(k)
            km.fit(randomReference)

            refDisp = km.inertia_
            refDisps[i] = refDisp
# Fit cluster to original data and create dispersion
        km = KMeans(k)
        km.fit(data)

        origDisp = km.inertia_
# Calculate gap statistic
        gap = np.log(np.mean(refDisps)) - np.log(origDisp)
# Assign this loop's gap statistic to gaps
        gaps[gap_index] = gap

        resultsdf = resultsdf.append({'clusterCount':k, 'gap':gap},
ignore_index=True)
    return (gaps.argmax() + 1, resultsdf)
score_g, df = optimalK(df, nrefs=5, maxClusters=10)
plt.plot(df['clusterCount'], df['gap'], linestyle='--', marker='o',
color='b');
plt.xlabel('K');
plt.ylabel('Gap Statistic');
plt.title('Gap Statistic vs. K');

# %%
kmeanModel = KMeans(n_clusters=3)
kmeanModel.fit(df)
centers = kmeanModel.cluster_centers_
print(centers)

# %%
plt.scatter(df.x,df.y)
for i in range(3):
    plt.scatter(centers[i,0], centers[i,1], s=200, c='r', marker='s')
```