Trabalho 02: Aprendizado Não Supervisionado

Universidade Estadual de Campinas (UNICAMP), Instituto de Computação (IC)

Prof. Jacques Wainer, 2021s2

Aluno: Maurício Pereira Lopes - RA: 225242

In [1]:
```
# carregamento de bibliotecas
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt

# configurações
random_state = 42
```

Leia este dataset que é um dataset de 1000 dados com 13 atributos. Os dados nao precisam ser normalizados.

Fonte do arquivo de dados: https://www.ic.unicamp.br/~wainer/cursos/2s2021/433/ex2-data.csv (https://www.ic.unicamp.br/~wainer/cursos/2s2021/433/ex2-data.csv)

In [3]:
```
# Carregar dados como um Pandas DataFrame e converter para um numpy array
url = "https://www.ic.unicamp.br/~wainer/cursos/2s2021/433/ex2-data.csv"
df = pd.read_csv(url, header = None, sep = r"\s+")
data = df.to_numpy()
data
```

Out[3]:
```
array([[-10.63,  -3.91,  27.69, ...,   5.32,   1.35,  -4.56],
       [ 12.56, -19.5 ,   4.39, ...,  -3.41,  -1.85,   2.55],
       [  4.19, -12.3 , -22.25, ...,   5.5 ,  11.08,  -2.19],
       ...,
       [  9.61, -21.71,  10.19, ...,  -2.  ,   2.94,   7.31],
       [-14.32,  -4.46,  -4.53, ...,  -3.52,  -1.13,   5.46],
       [ 17.4 ,  11.46,  14.03, ...,  -3.86,  -4.54,   3.02]])
```

1- K-Means

- Rode o K-Means com k de 2 a 15

In [4]:
```
# Definir uma lista com os valores de k indo de 2 a 15
k_list = list(range(2, 16, 1))
kmeans = {}
for k in k_list:
    kmeans[k] = KMeans(n_clusters = k, random_state = random_state).fit(data)
```

- Use silhueta e pelo menos alguma outra medida interna de qualidade

In [5]:
```
# Silhouette Coefficient
silhouette = {}
for k in k_list:
    labels = kmeans[k].labels_
    silhouette[k] = metrics.silhouette_score(data, labels, metric = 'euclidean')
```

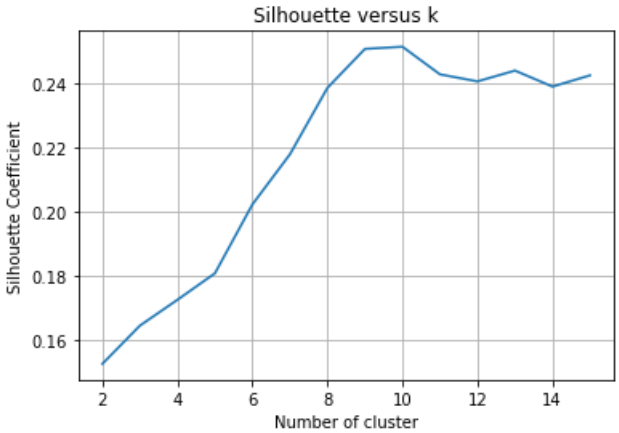
In [6]:
```python
# Calinski-Harabasz Index
calinski = {}
for k in k_list:
    labels = kmeans[k].labels_
    calinski[k] = metrics.calinski_harabasz_score(data, labels)
```

In [7]:
```python
# Davies-Bouldin Index
davies = {}
for k in k_list:
    labels = kmeans[k].labels_
    davies[k] = metrics.davies_bouldin_score(data, labels)
```

In [8]:
```python
# Sum of Squared Errors (SSE)
sse = {}
for k in k_list:
    # Inertia: Sum of distances of samples to their closest cluster center
    sse[k] = kmeans[k].inertia_
```

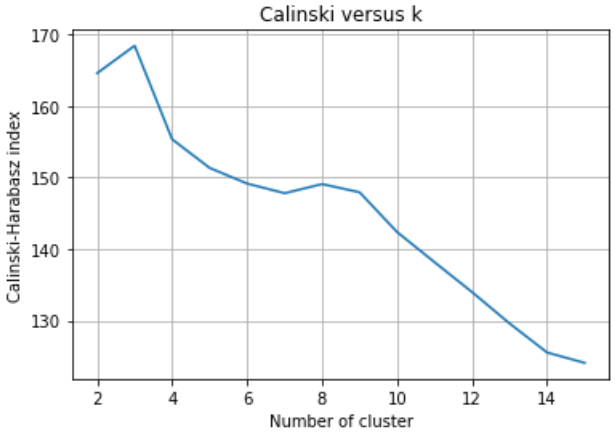
- Mostre o gráfico dessas medidas versus o k

In [9]:
```python
# Silhouette versus k
'''
Mais próximo de 1 indica clusterização mais densa.
Valores perto de zero indicam sobreposição entre clusters.
'''
plt.plot(list(silhouette.keys()), list(silhouette.values()), label = 'linear')
plt.grid()
plt.title("Silhouette versus k")
plt.xlabel("Number of cluster")
plt.ylabel("Silhouette Coefficient")
plt.show()
```



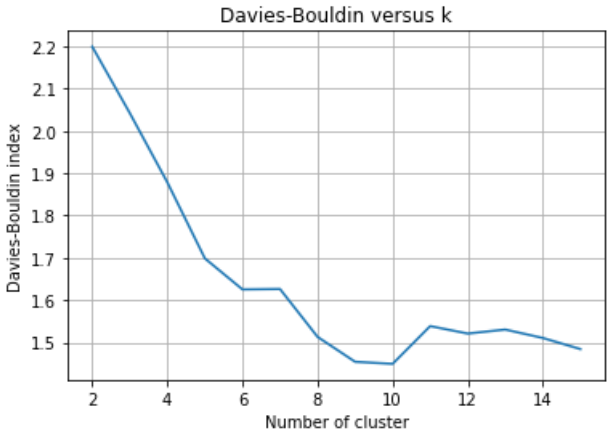
In [10]:

```
# Calinski-Harabasz versus k
'''
O valor é alto quando os clusters são densos e
bem separados.
'''
plt.plot(list(calinski.keys()), list(calinski.values()), label = 'linear')
plt.grid()
plt.title("Calinski versus k")
plt.xlabel("Number of cluster")
plt.ylabel("Calinski-Harabasz index")
plt.show()
```

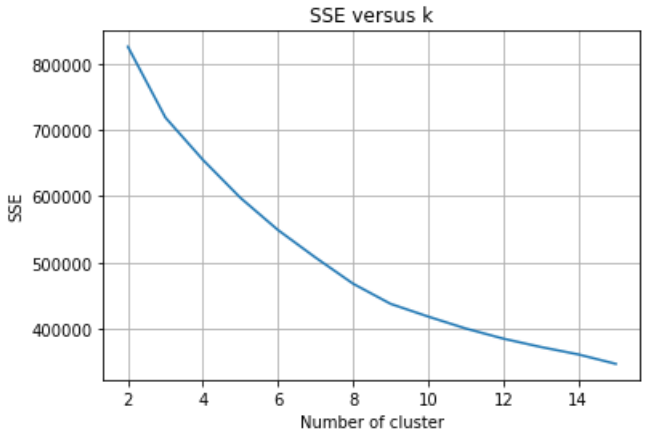


In [11]:

```
# Davies-Bouldin versus k
'''
Zero é o menor valor possível.
Valores próximos de zero indicam melhor clusterização.
'''
plt.figure()
plt.plot(list(davies.keys()), list(davies.values()))
plt.grid()
plt.title("Davies-Bouldin versus k")
plt.xlabel("Number of cluster")
plt.ylabel("Davies-Bouldin index")
plt.show()
```



In [12]:
```
# Sum of Squared Errors (SSE) versus k
'''
Estamos buscando por um cotovelo (elbow) no gráfico,
a partir do qual, não haveria ganho significativo na
redução da soma dos erros quadrados ao se aumentar o k.
'''
plt.figure()
plt.plot(list(sse.keys()), list(sse.values()))
plt.grid()
plt.title("SSE versus k")
plt.xlabel("Number of cluster")
plt.ylabel("SSE")
plt.show()
```



2 - Escolha um k

Analisando os gráficos das métricas acima, um k razoável poderia ser do 8 ao 10. Nesta feixa vemos o seguinte comportamento nos gráficos:

- Maior coeficiente de silhueta
- Valor intermadiário no índice de Calinski-Harabasz
- Valor bem próximo ao mínimo do índice de Davies-Bouldin
- SSE encaminhando para a estabilidade

Estou optando por um k = 9.

In [13]:
```
# atribuindo o valor escolhido para k
k = 9
# obtendo os labels de cada dado conforme a clusterização com o k escolhido
labels_kmeans = kmeans[k].labels_
```

3 - GMM

Usando o k escolhido no item anterior, rode o GMM

- Com gaussianas esféricas (matrizes de covariancia sao a matriz identidade vezes uma constante)

In [14]:
```
labels_gmm_esf = GaussianMixture(n_components = k, covariance_type = "spherical",
                                 random_state = random_state).fit_predict(data)
```

- Com gaussianas diagonais (matrizes de covariancia sao matrizes diagonais)

In [15]:
```
labels_gmm_diag = GaussianMixture(n_components = k, covariance_type = "diag",
                                  random_state = random_state).fit_predict(data)
```

- Com gaussianas sem restrição (as matrizes de covariancia são livres)

In [16]:
```
labels_gmm_full = GaussianMixture(n_components = k, covariance_type = "full",
                                  random_state = random_state).fit_predict(data)
```

4 - Medidas externas para comparar duas clusterizações

- Use pelo menos 2 medidas externas para comparar a solução do GMM sem restrição (full) com as outras 2 (esférica e diagonal).

In [17]:
```
# Rand index
rand_full_esf = metrics.rand_score(labels_gmm_full, labels_gmm_esf)
rand_full_diag = metrics.rand_score(labels_gmm_full, labels_gmm_diag)
print("Rand index entre\nGMM Full e GMM Esférico:", rand_full_esf)
print("GMM Full e GMM Diagonal:", rand_full_diag)
```

```
Rand index entre
GMM Full e GMM Esférico: 0.9323103103103103
GMM Full e GMM Diagonal: 0.9464984984984985
```

In [18]:
```
# Mutual Information based scores
mutual_full_esf = metrics.adjusted_mutual_info_score(labels_gmm_full, labels_gmm_esf)
mutual_full_diag = metrics.adjusted_mutual_info_score(labels_gmm_full, labels_gmm_diag)
print("Mutual information score entre\nGMM Full e GMM Esférico:", mutual_full_esf)
print("GMM Full e GMM Diagonal:", mutual_full_diag)
```

```
Mutual information score entre
GMM Full e GMM Esférico: 0.7873659750308852
GMM Full e GMM Diagonal: 0.7966804927436857
```

In [19]:
```
# Fowlkes-Mallows scores
fms_full_esf = metrics.fowlkes_mallows_score(labels_gmm_full, labels_gmm_esf)
fms_full_diag = metrics.fowlkes_mallows_score(labels_gmm_full, labels_gmm_diag)
print("Fowlkes-Mallows score entre\nGMM Full e GMM Esférico:", fms_full_esf)
print("GMM Full e GMM Diagonal:", fms_full_diag)
```

```
Fowlkes-Mallows score entre
GMM Full e GMM Esférico: 0.748992634365838
GMM Full e GMM Diagonal: 0.7769945956528681
```

Como uma comparação adicional, não solicitada no enunciado, foi calculado o Rand index entre GMM Full e KMeans com o k escolhido. Pode-se ver que, pelo Rand Index, o KMeans chegou a um resultado muito próximo ao GMM_Full.

In [20]:
```
# Rand index
rand_full_kmeans = metrics.rand_score(labels_gmm_full, labels_kmeans)
print("Rand index entre\nGMM Full e KMeans:", rand_full_kmeans)
```

```
Rand index entre
GMM Full e KMeans: 0.9461161161161161
```

Em todas as métricas externas calculadas acima, os melhores resultados foram entre GMM_Full e GMM_Diagonal. Isso indica que estes dois modelos geraram cluesterizações mais parecidas. Se os labels obtidos com o GMM_Full fossem considerados como Ground Truth, o GMM_Diag teria sido o modelo com melhor resultado de clusterização.

In [ ]:
```
```