Семинарски рад у оквиру курса Истраживање података 2

Кластеровање аеродрома на основу географске локације, повезаности летовима и узимајући у обзир додатне транспортне могућности

Професор : Ненад Митић
Студент : Чолић Ања 231/2019

Садржај

Сажетак

Рад се заснива на кластеровању података о аеродромима и летовима који су расположиви на https://openflights.org/data.html на основу географцке локације, повезаности летовима и узимајући у обзир додатне транспортне могућности (воз, трајект, ...).
Резултати анализе примене алгоритама кластеровања су приказани како текстуално тако и графички у програмском језику пајтон.

Описи алгоритама

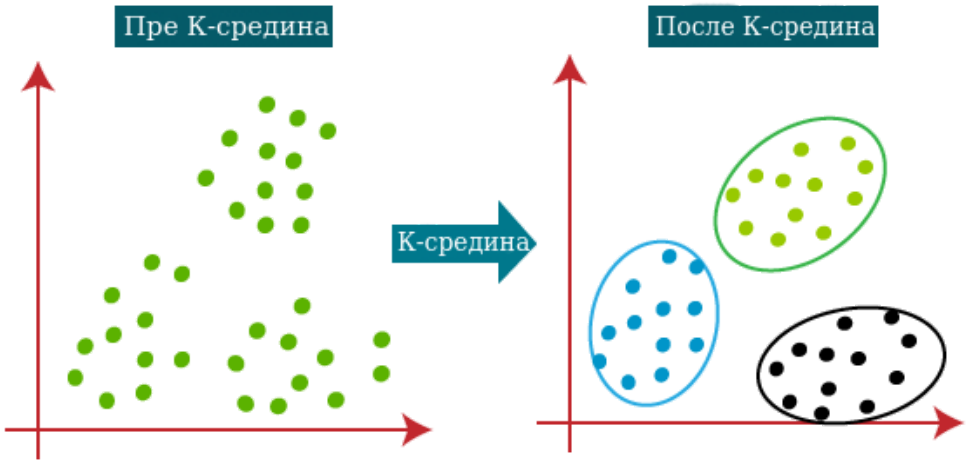
К-средина

К-средина је алгоритам кластеровања који се користи за груписање сличних тачака података у одређени број кластера. Ради тако што прво иницијализује случајне позиције центара кластера у простору података, а затим итеративно покушава да минимизује суму квадратних удаљености између тачака података и центара кластера.

Поступак се обично одвија на следећи начин:

1. Иницијализуј центре кластера тако што изабереш случајне тачке података као почетне позиције центара.
2. Сваку тачку података додели кластеру који је најближи по удаљености од центара кластера.
3. Израчунај нове позиције центара кластера тако што се просечне вредности тачака у сваком кластеру користе као нове позиције центара.
4. Понови кораке 2 и 3 све док се не достигне одређени број итерација или се не постигне конвергенција.

На крају, добијају се кластери који садрже сличне тачке података, а тачке унутар једног кластера су међусобно сличније него са тачкама из других кластера. К-средина алгоритам се поред истраживања података, обично користи још и у областима као што су анализа података и машинско учење.



Агломеративно кластеровање (Agglomerative Clustering)

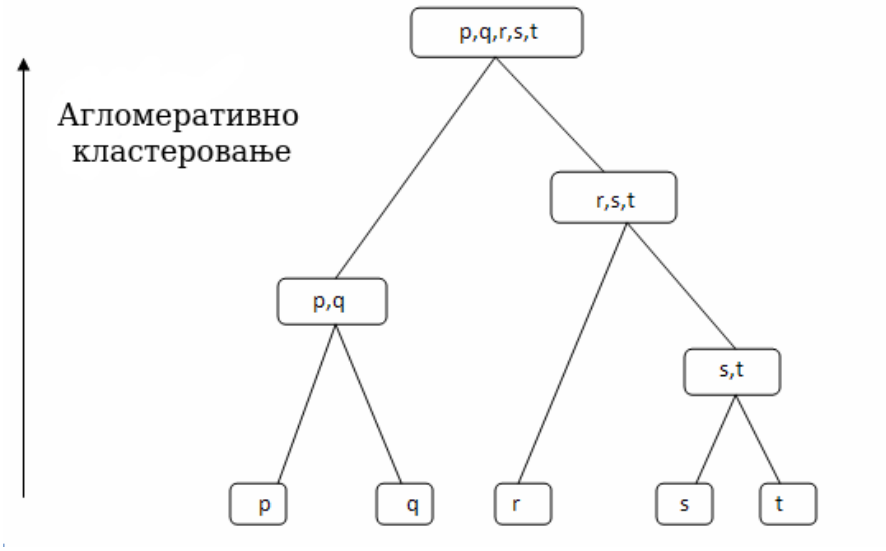
Агломеративно кластеровање (Agglomerative Clustering) је хијерархијско кластеровање у коме се сваки објекат најпре сматра јединичним кластером, а затим се они постепено спајају у веће кластере до крајњег формирања једног великог кластера који садржи све податке. При томе, спајање кластера се врши на основу сличности између њих.

1. Први корак је одређивање удаљености између сваког пара инстанци.
2. Затим се свака инстанца сматра кластером, тако да се добијају N кластера, где је N број инстанци.
3. Следећи корак је формирање дендрограма. У овом кораку, одређује се растојање између кластера. То се ради тако што се прво одреди растојање између најближих кластера, затим се

два кластера спајају у један и поново се одређује растојање између новог кластера и сваког преосталог кластера. Поступак се понавља док се не добије један кластер који садржи све инстанце.

4. На основу дендрограма, одлучује се на колико кластера треба поделити податке. Ова одлука се доноси на основу растојања између кластера.
5. На крају, кластери се формирају спајањем најближих кластера на основу хијерархије коју је дефинисао дендрограм.

Постоје различите могућности за спајање кластера, такозване мере сличности (linkage methods), као што су Ward, Complete, Average, и Single linkage. Агломеративно кластеровање често се користи и у областима као што су биологија, медицина, економија, маркетинг, итд.



DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN је алгоритам за кластеровање података који се заснива на густини података. Овај алгоритам може да открије произвољно обликоване кластере у подацима и идентификује изоловане тачке.

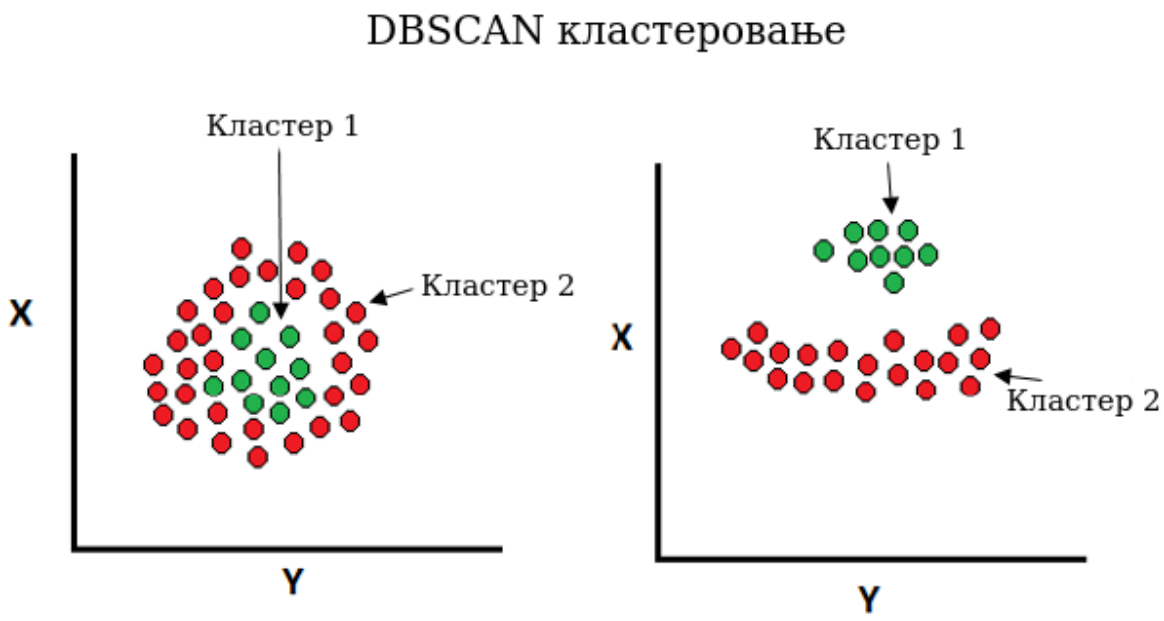
Главна предност DBSCAN алгоритма је што не захтева претходно одређивање броја кластера у подацима. Међутим, потребно је одређивати параметре алгоритма како би се добили најбољи резултати.

Кораци DBSCAN алгоритма су:

1. Одређивање параметара: Потребно је одредити два кључна параметра, а то су епсилон радијус eps и минимални број података у групи min_samples.
2. Проналажење суседа: За сваки податак у скупу података, одређује се број суседа који се налазе унутар епсилон радијуса.
3. Формирање кластера: Ако број суседа унутар епсилон радијуса података прелази min_samples, подаци се сматрају делом групе. Нове тачке које се налазе унутар епсилон радијуса тих тачака такође се додају групи. Овај поступак се наставља док се сви подаци који могу бити додати групи не додају.
4. Идентификација изоловане тачке: Тачке које се не налазе у групи, а не задовољавају услове за формирање нове групе, сматрају се изолованим или шумом.

5. Обрада група: Након формирања класетра, постоји могућност да се два или више класетра преклапају. DBSCAN алгоритам може да реши ове преклапања тако што ће спојити блиске класетре у један, или их одвојити у два.
6. Излаз: DBSCAN алгоритам враћа информације о групама, као и о изолованим тачкама у скупу података.

Овај алгоритам је популаран за груписање података у областима као што су геоинформатика, машинско учење, препознавање образаца и друге сличне области.



Опис података

Подаци који су коришћени за кластеровање су преузети са OpenFlights веб сајта, који пружа бесплатне информације о аеродромима, авио-компанијама, рути летова и другим аспектима авио-индустрије.
Конкретно у овом случају, скупови података који су нам били од значаја су следећи:

1. **Скуп података "airports.dat":**

- **Airport ID**: Јединствени идентификациони број аеродрома.
- **Name**: Назив аеродрома.
- **City**: Град у ком се аеродром налази.
- **Country**: Држава у ком се аеродром налази.
- **IATA**: Трокарактерни код за идентификацију аеродрома, који се користи за једноставно препознавање и претрагу аеродрома.
- **ICAO**: Четвороцифрени код за идентификацију аеродрома, који се користи у авио-саобраћају и ваздухопловним картама.
- **Latitude**: Географска ширина аеродрома у децималном запису.
- **Longitude**: Географска дужина аеродрома у децималном запису.
- **Altitude**: Надморска висина аеродрома у метрима.

	Airport ID	Name	City	Country	IATA	ICAO	Latitude	Longitude	Altitude	Timezone	DST	Tz database time zone	Type	Source
0	1	Goroka Airport	Goroka	Papua New Guinea	GKA	AYGA	-6.081690	145.391998	5282	10	U	Pacific/Port_Moresby	airport	OurAirports
1	2	Madang Airport	Madang	Papua New Guinea	MAG	AYMD	-5.207080	145.789001	20	10	U	Pacific/Port_Moresby	airport	OurAirports
2	3	Mount Hagen Kagamuga Airport	Mount Hagen	Papua New Guinea	HGU	AYMH	-5.826790	144.296005	5388	10	U	Pacific/Port_Moresby	airport	OurAirports
3	4	Nadzab Airport	Nadzab	Papua New Guinea	LAE	AYNZ	-6.569803	146.725977	239	10	U	Pacific/Port_Moresby	airport	OurAirports
4	5	Port Moresby Jacksons International Airport	Port Moresby	Papua New Guinea	POM	AYPY	-9.443380	147.220001	146	10	U	Pacific/Port_Moresby	airport	OurAirports

2. **Скуп података "routes.dat":** Овај скуп података садржи информације о рутама летова између различитих аеродрома широм света. Сваки ред у овој табели представља једну руту лета и садржи следеће атрибуте:

- **Airline**: Код авио-компаније која обавља лет.
- **Airline ID**: Јединствени идентификатор авио-компаније која обавља лет.
- **Source airport**: ИАТА код полазног аеродрома.
- **Source airport ID**: Јединствени идентификатор полазног аеродрома.
- **Destination airport**: ИАТА код долазног аеродрома.
- **Destination airport ID**: Јединствени идентификатор долазног аеродрома.
- **Codeshare**: Да ли се лет дели са другом авио-компанијом.
- **Stops**: Број преседања на лету.
- **Equipment**: Опрема која се користи.

	Airline	Airline ID	Source airport	Source airport ID	Destination airport	Destination airport ID	Codeshare	Stops	Equipment
0	2B	410	AER	2965	KZN	2990	NaN	0	CR2
1	2B	410	ASF	2966	KZN	2990	NaN	0	CR2
2	2B	410	ASF	2966	MRV	2962	NaN	0	CR2
3	2B	410	CEK	2968	KZN	2990	NaN	0	CR2
4	2B	410	CEK	2968	OVB	4078	NaN	0	CR2

3. **Скуп података "airports-extended.dat":** Овај скуп података пружа проширене информације о аеродромима.
    - Атрибут **Type** аеродрома је једини атрибут који се у овом скупу података разликује од **"airports.dat"**. Вредност „аеродром“ (airport) је за аеродромске терминале, „станица“ (station) за железничке станице, „лука“ (port) за трајектне терминале и „непознато“ ако није познато. У **"airports.dat"** је укључен само Type=airport.

Кластеровање аеродрома на основу географске локације

Препроцесирање

Импортовање свих потребних библиотека

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import pairwise_distances
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score
```

Учитавање података о аеродромима из датотеке **airports.dat**

```
airports_url = 'https://raw.githubusercontent.com/jpatokal/openflights/master/data/airports.dat'
airports = pd.read_csv(airports_url, header=None, names=['Airport ID', 'Name', 'City', 'Country', 'IATA', 'ICAO', 'La
```

Уклањање недостајућих вредности, дупликата и конверзија типа података

```
airports.dropna()
airports.drop_duplicates()
airports['Latitude'] = airports['Latitude'].astype(float)
airports['Longitude'] = airports['Longitude'].astype(float)
```

Идеја решења и имплементација алгоритма

Дакле, предстојећи део имплементације се бави самим кластеровањем аеродрома на основу њихове географске локације (дужине и ширине). Идеја је да се аеродроми групишу у кластере на основу њихове сличности у погледу географске локације, што би могло бити корисно за различите анализе и планирање летова.
Следећи корак је креирање матрице растојања између аеродрома. Ова матрица се рачуна помоћу **pairwise_distances** функције из **sklearn.metrics.pairwise**. Функција као улазни параметар узима низ вредности којима су представљене координате свих аеродрома (географску ширину и дужину) и метрику растојања која се користи за рачунање удаљености између аеродрома. У овом случају, користи се еуклидско растојање.

```
distance_matrix = pairwise_distances(
airports[['Latitude', 'Longitude']].values, metric='euclidean')
```

```
distance_matrix
```

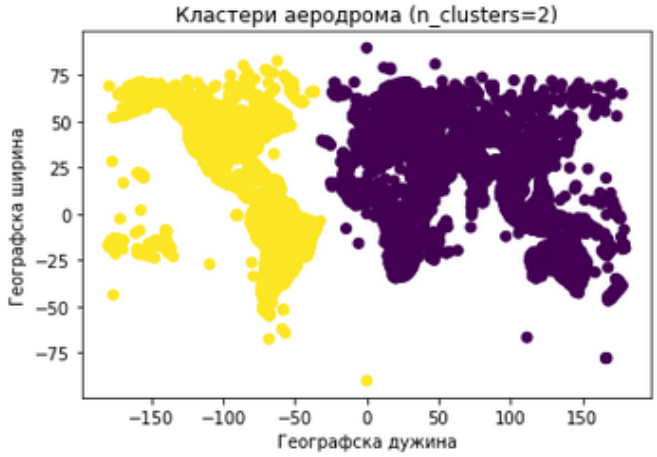
```
array([[  0.        ,   0.96049679,   1.12524431, ..., 131.0898597 ,
        217.20625595, 122.16418415],
       [  0.96049679,   0.        ,   1.61650182, ..., 131.00690792,
        217.68823831, 122.14654549],
       [  1.12524431,   1.61650182,   0.        , ..., 130.01126209,
        216.14064138, 121.06628361],
       ...,
       [131.0898597 , 131.00690792, 130.01126209, ...,   0.        ,
        133.46895774,  12.38189807],
       [217.20625595, 217.68823831, 216.14064138, ..., 133.46895774,
          0.        , 129.42478387],
       [122.16418415, 122.14654549, 121.06628361, ...,  12.38189807,
        129.42478387,   0.        ]])
```

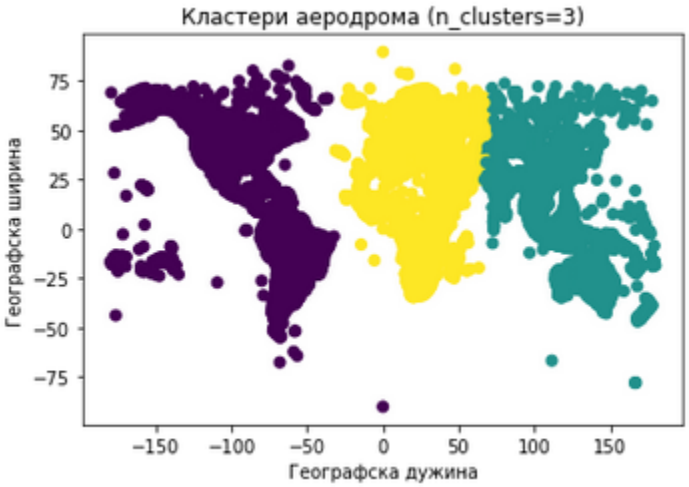
Након тога, дефинише се функција **plot_clusters** која прима два улазна параметра: n_clusters и distance_matrix. Ова функција **врши кластеровање података коришћењем хијерархијског агломеративног кластеровања са ward-овом методом повезивања**, а број кластера одговара вредности параметра

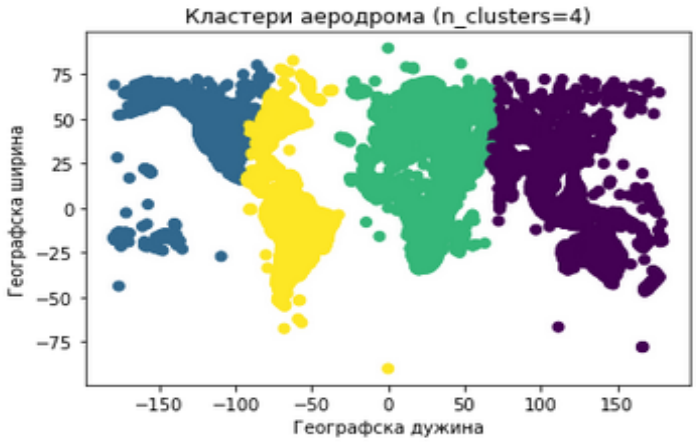
n_clusters(од 2 до 10). Резултујући кластери се визуализују помоћу matplotlib библиотеке и scatter плотова.

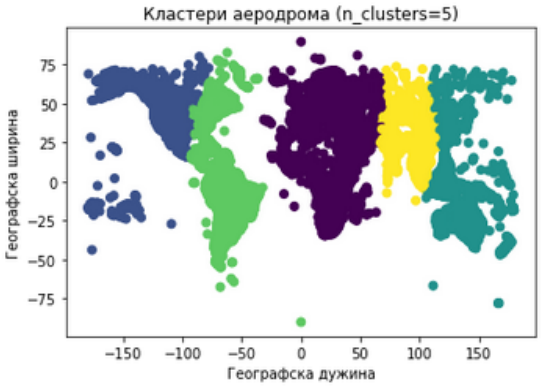
```
def plot_clusters(n_clusters, distance_matrix):
    agg_clustering = AgglomerativeClustering(n_clusters=n_clusters, linkage='ward')
    labels = agg_clustering.fit_predict(distance_matrix)
    plt.scatter(x=airports['Longitude'], y=airports['Latitude'], c=labels, cmap='viridis')
    plt.xlabel('Географска дужина')
    plt.ylabel('Географска ширина')
    plt.title('Кластери аеродрома (n_clusters={})'.format(n_clusters))
    plt.show()

for n_clusters in [2, 3, 4, 5, 6, 7, 8, 9, 10]:
    plot_clusters(n_clusters, distance_matrix)
```

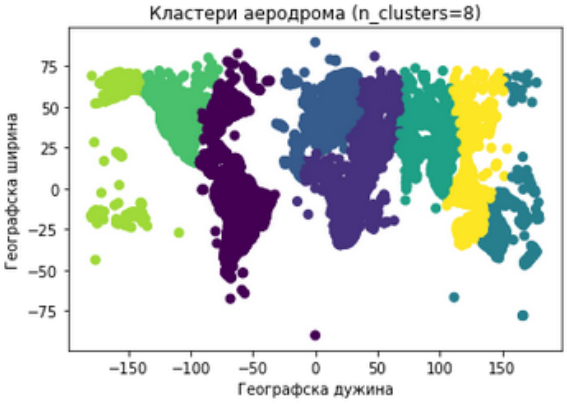
Приказивање scatter плотова који приказују расподелу аеродрома по кластерима за различите бројеве кластера, од два до десет:

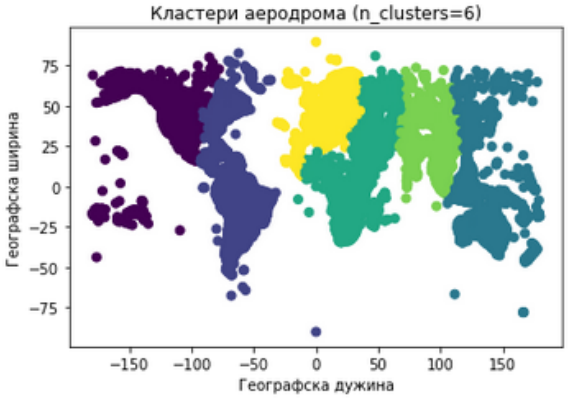


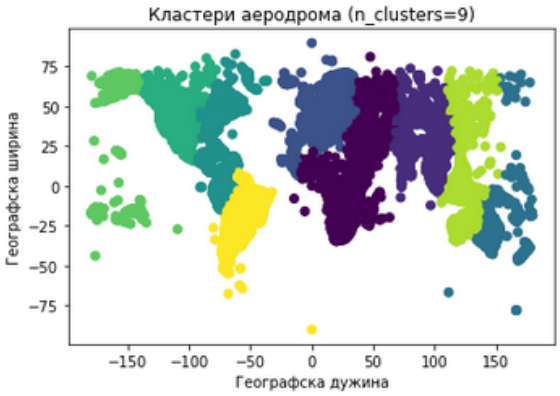


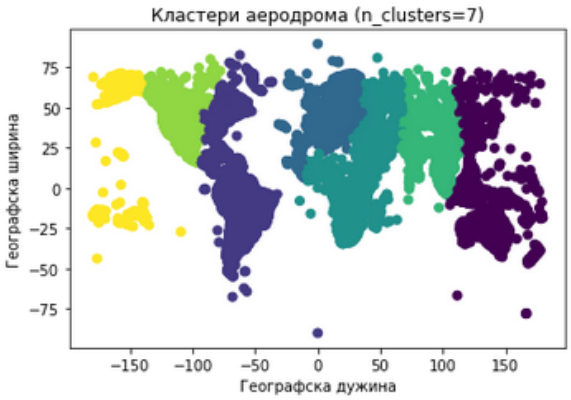

Кластери аеродрома (n_clusters=4)
Географска ширина
Географска дужина
75
50
25
0
−25
−50
−75
−150
−100
−50
0
50
100
150

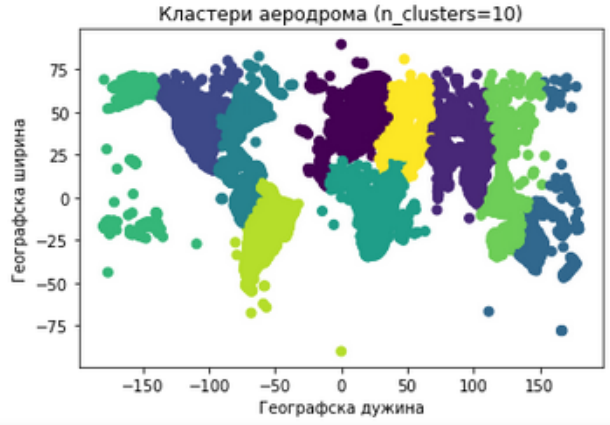

Кластери аеродрома (n_clusters=5)
Географска ширина
Географска дужина
75
50
25
0
−25
−50
−75
−150
−100
−50
0
50
100
150


Кластери аеродрома (n_clusters=8)
Географска ширина
Географска дужина
75
50
25
0
−25
−50
−75
−150
−100
−50
0
50
100
150


Кластери аеродрома (n_clusters=6)
Географска ширина
Географска дужина
75
50
25
0
−25
−50
−75
−150
−100
−50
0
50
100
150


Кластери аеродрома (n_clusters=9)
Географска ширина
Географска дужина
75
50
25
0
−25
−50
−75
−150
−100
−50
0
50
100
150





Процена квалитета решења

Како би се одредио оптималан број кластера и оценио квалитет кластеровања, поред претходне визуелизације, у наставку кода се користе и три различите метрике:

1. Silhouette скор (Коефицијент силуете) се креће од -1 до 1, где већи скорови указују на бољу поделу података на кластере. Рачуна се као разлика између просечне дистанце до других објеката у истом кластеру и просечне дистанце до објеката у најближем суседном кластеру. Већа вредност коефицијента силуете указује на то да је објекат добро сврстан у свој кластер и да се не меша са објектима из других кластера.

2. Calinski-Harabasz скор покушава да измери колико су кластери добро дефинисани, односно колико је велика разлика између средњих вредности објеката унутар кластера и између кластера. Овај скор се рачуна као однос суме квадрата

међукластер варијансе и суме квадрата унутар кластера. Већи скорови указују на бољу поделу података на кластера.

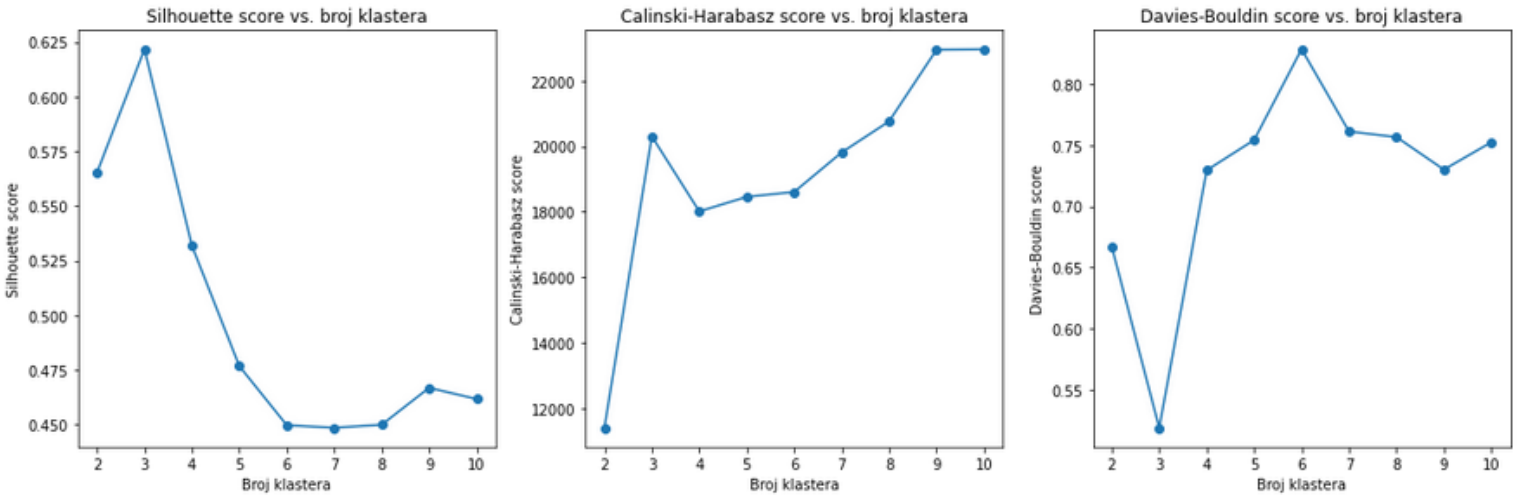
3. Davies-Bouldin скор се такође користи за мерење квалитета кластеровања али покушава да измери колико се кластери разликују. Рачуна се као просечна вредност индекса[1] сличности између сваког кластера и његовог најсличнијег кластера. Нижи скорови указују на бољу поделу података на кластера.

Дакле, рачунамо за сваки број кластера у опсегу од 2 до 10. Циљ је одабрати број кластера који максимизује коефицијент силуете и Calinski-Harabasz скор, а минимизује Davies-Bouldin скор.

```
n_clusters_range = range(2, 11)
silhouette_scores2 = []
calinski_harabasz_scores = []
davies_bouldin_scores = []
for n_clusters in n_clusters_range:
    agg_clustering = AgglomerativeClustering(n_clusters=n_clusters, linkage='ward')
    labels = agg_clustering.fit_predict(distance_matrix)
    silhouette_scores2.append(silhouette_score(distance_matrix, labels))
    calinski_harabasz_scores.append(calinski_harabasz_score(distance_matrix, labels))
    davies_bouldin_scores.append(davies_bouldin_score(distance_matrix, labels))

fig, axs = plt.subplots(1, 3, figsize=(15, 5))
axs[0].plot(n_clusters_range, silhouette_scores2, marker='o')
axs[0].set_xlabel('Broj klastera')
axs[0].set_ylabel('Silhouette score')
axs[0].set_title('Silhouette score vs. broj klastera')

axs[1].plot(n_clusters_range, calinski_harabasz_scores, marker='o')
axs[1].set_xlabel('Broj klastera')
axs[1].set_ylabel('Calinski-Harabasz score')
axs[1].set_title('Calinski-Harabasz score vs. broj klastera')

axs[2].plot(n_clusters_range, davies_bouldin_scores, marker='o')
axs[2].set_xlabel('Broj klastera')
axs[2].set_ylabel('Davies-Bouldin score')
axs[2].set_title('Davies-Bouldin score vs. broj klastera')

plt.tight_layout()
plt.show()
```

Резултати су визуелизовани у три графа како би се олакшало поређење метрика.

[1] Индекс сличности се рачуна као однос унутар кластер варијансе и растојања између центроида кластера.



Кластеровање аеродрома на основу повезаности летовима

Препроцесирање

Импортовање свих потребних библиотека

```
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
```

Учитавање података о аеродромима из датотеке **airports.dat,** као и података о рутама из датотеке **routes.dat**

```
airports = pd.read_csv('airports.dat', header=None, names=['AirportID', 'Name', 'City', 'Country', 'IATA', 'ICAO', 'La
routes = pd.read_csv('routes.dat', header=None, names=['Airline', 'AirlineID', 'SourceAirport', 'SourceAirportID', 'De
```

Идеја решења и имплементација алгоритма

Предстојећи део имплементације се бави самим кластеровањем аеродрома на основу повезаности летовима. Главна идеја овог решења је да се креира граф који представља мрежу летова између различитих аеродрома.

Дакле, креирамо празни граф користећи **nx.Graph()**. Ово представља основу наше мреже.
Затим пролазимо кроз све аеродроме и за сваки аеродром додајемо чвор у граф.
Пролазимо и кроз све руте и за сваку руту проверавамо да ли постоје одговарајући аеродроми за изворни и одредишни аеродром. Ако су одговарајући аеродроми пронађени, додајемо грану (везу) између изворног и одредишног аеродрома у графу.
Након што смо додали све чворове и гране у граф, користимо **nx.to_numpy_array(G)** да бисмо креирали матрицу повезаности. Редови и колоне одговарају чворовима (аеродромима), а елементи матрице представљају везе између аеродрома (1 ако постоји веза, 0 ако не постоји веза).

```
# Креирање графа
G = nx.Graph()

# Додавање чворова за сваки аердором
for _, airport in airports.iterrows():
    G.add_node(airport['IATA'], Name=airport['Name'], City=airport['City'], Country=airport['Country'], Latitude=airp

# Додавање грана на основу рута
for _, route in routes.iterrows():
    source_airport_data = airports[airports['IATA'] == route['SourceAirport']]
    dest_airport_data = airports[airports['IATA'] == route['DestinationAirport']]

    # Провера да ли постоји аеродром
    if not source_airport_data.empty and not dest_airport_data.empty:
        source_airport = source_airport_data['IATA'].values[0]
        dest_airport = dest_airport_data['IATA'].values[0]
        G.add_edge(source_airport, dest_airport)

# Креирање матрице повезаности
adjacency_matrix = nx.to_numpy_array(G)
```

Добијена је матрица повезаности облика:

```
array([[0., 1., 1., ..., 0., 0., 0.],
       [1., 0., 1., ..., 0., 0., 0.],
       [1., 1., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

Коначно примењујемо алгоритам DBSCAN на стандардизоване податке. Овај алгоритам је непараметарски (не захтева унапред познат број кластера) и може аутоматски одредити број кластера на основу густине података. Параметри **eps** и **min_samples** су кључни за рад DBSCAN-а, да поновимо:

- **eps** је радијус околине око сваког тачкастог податка. Ако има довољно тачака унутар овог радијуса, они се сматрају делом истог кластера.
- **min_samples** је минимални број тачака који треба да буде унутар **eps** радијуса како би се формирао кластер.

У овом коду, DBSCAN се примењује са параметрима **eps=1.2** и **min_samples =10**. То значи да ће алгоритам тражити кластер који има најмање 10 тачака унутар радијуса од 1.2 стандардне девијације. Резултат овог корака је низ cluster_labels који садржи ознаке кластера којима припадају аеродроми.

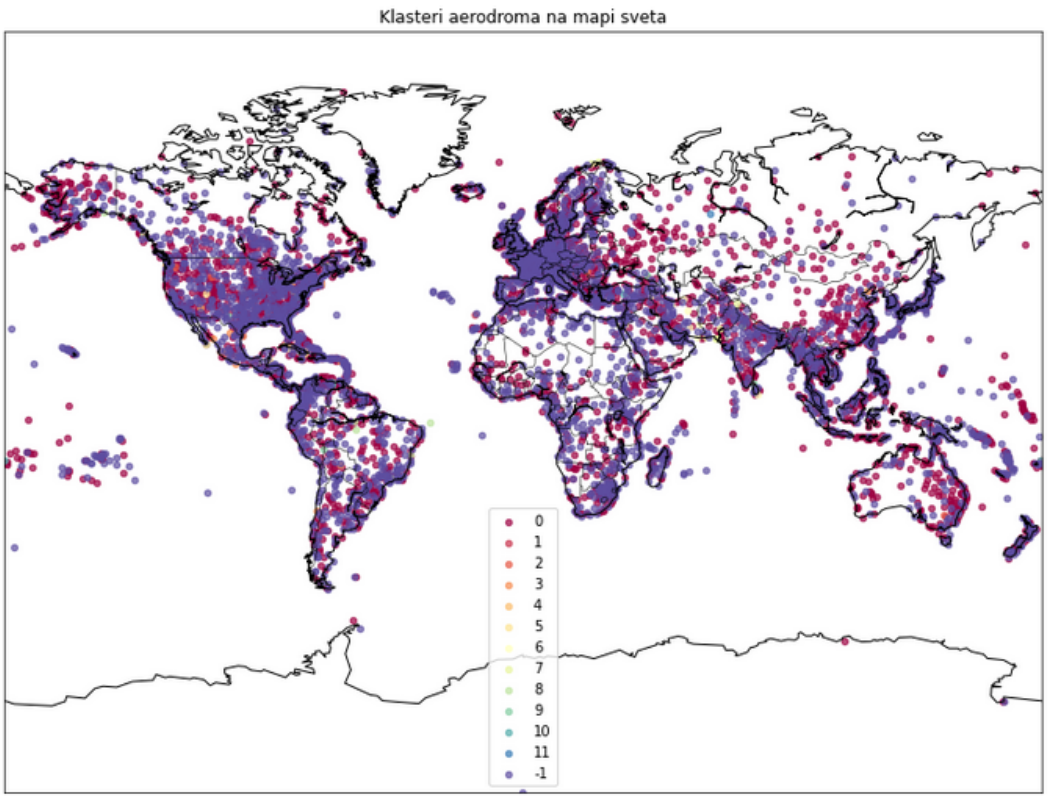
DBSCAN ће означити тачке које нису део ни једног кластера као "шум" (noise), што значи да оне нису јасно повезане ни са једним кластером.

```
# Стандардизација података
scaler = StandardScaler()
adjacency_matrix_scaled = scaler.fit_transform(adjacency_matrix)

# DBSCAN
dbscan = DBSCAN(eps=1.2, min_samples=10, metric='euclidean')
cluster_labels = dbscan.fit_predict(adjacency_matrix_scaled)
```

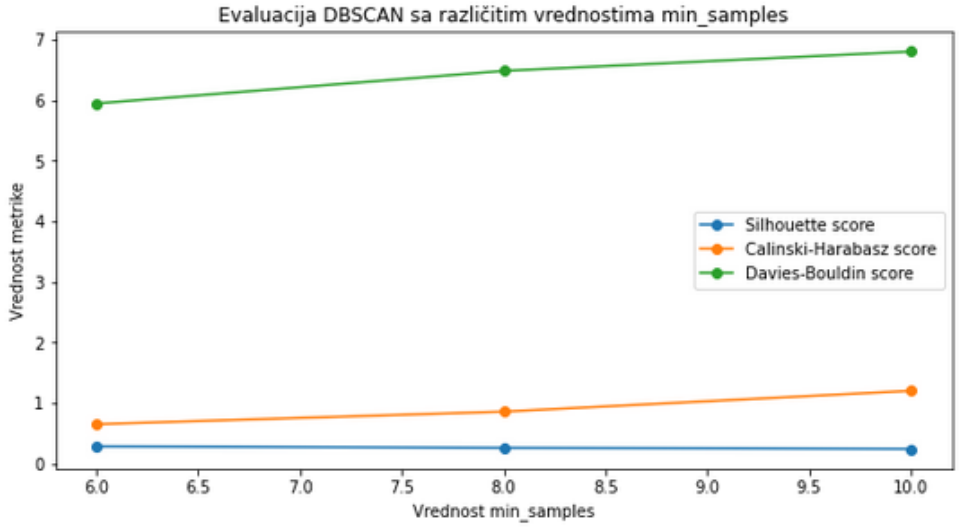
Визуелизација и процена квалитета решења

Резултујући кластери се визуализују помоћу matplotlib библиотеке и Basemap објекта. Свака тачка на мапи представља један аеродром. Боје тачака означавају различите кластере аеродрома. Међутим, постоје и неке тачке са вредношћу -1. Аеродроми који су означени са -1 не припадају ниједном од дефинисаних кластера и сматрају се "шумом".

```
# Визуелизација на мапи света
plt.figure(figsize=(16, 10))
m = Basemap(projection='mill', resolution='c', lon_0=0)
m.drawcoastlines()
m.drawcountries()


x, y = m(list(airports['Longitude']), list(airports['Latitude']))


unique_labels = set(cluster_labels)
colors = plt.cm.Spectral(np.linspace(0, 1, len(unique_labels)))
for label, color in zip(unique_labels, colors):
    cluster_mask = cluster_labels == label
    cluster_x = [x[i] for i, mask in enumerate(cluster_mask) if mask]
    cluster_y = [y[i] for i, mask in enumerate(cluster_mask) if mask]
    m.scatter(cluster_x, cluster_y, s=20, c=color, label=label, alpha=0.7)

plt.legend()
plt.title('Klasteri aerodroma na mapi sveta')
plt.show()
```




```
min_samples=10, broj klastera=13
min_samples=8, broj klastera=21
min_samples=6, broj klastera=35
```



Приметимо да при оваквом кластеровању неће бити елемената на граници. Аеродром који је повезан са бар још једним ући ће у кластер и биће део компоненте повезаности. Аеродроми који нису повезани ниједним летом припашће шуму.

Ова визуализација је коришћена за процену квалитета решења и како бисмо добили боље разумевање структуре мреже ваздушног саобраћаја.

Кластеровање аеродрома узимајући у обзир додатне транспортне могућности

Приликом кластеровања на основу географске локације, главни атрибути који се користе за груписање су географска ширина и дужина аеродрома. То може довести до кластера који су просторно блиски једни другима, али не узимају у обзир друге карактеристике аеродрома које се могу разликовати, као што су додатне транспортне могућности.

Уколико додамо и ово као значајан атрибут, можемо добити као резултат кластеровања кластере који садрже сличне типове аеродрома, без обзира на њихову географску близину. То може бити корисно ако желите да анализирате аеродроме са сличним функционалностима или карактеристикама, без обзира на њихову локацију.

Препроцесирање

Импортовање свих потребних библиотека

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
```

Учитавање података о аеродромима из датотеке **airports_extended.dat** и издвајање атрибута о географској локацији и типу

```
data = pd.read_csv('airports-extended.dat', header=None, names=['AirportID', 'Name', 'City', 'Country', 'IATA', 'ICAO

X = data[['Latitude', 'Longitude', 'Type']]
X = pd.get_dummies(X, columns=['Type'], prefix=['Type'])
```

Имплементација алгоритма

Прво, дефинишемо опсег броја кластера које желимо испробати. У овом случају користимо range (3, 11), што значи да ћемо тестирати број кластера од 3 до 10 укључујући 10.

Затим иницијализујемо празну листу силета коефицијената у којој ћемо чувати коефицијент силуете за сваки број кластера.

Унутар петље, креирамо K-средина модел са тренутном вредношћу броја кластера.

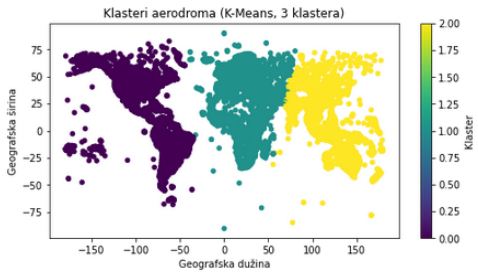
Извршавамо кластеризацију над подацима X и добијамо ознаке кластера за сваки аеродром у променљивој **cluster_labels**.

Након што завршимо све итерације за различите бројеве кластера, визуализујемо резултате, користимо **plt.show()** да прикажемо графикон.

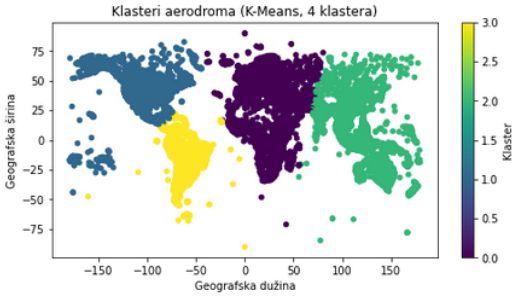
```
num_clusters = range(3, 11)
silhouette_scores = []


for n_clusters in num_clusters:
    kmeans = KMeans(n_clusters=n_clusters, random_state=0)
    cluster_labels = kmeans.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)
    silhouette_scores.append(silhouette_avg)


plt.plot(num_clusters, silhouette_scores, marker='o')
plt.xlabel('Broj klastera')
plt.ylabel('Silhouette Skor')
plt.title('Silhouette Skor za različite brojeve klastera')
plt.grid(True)
plt.show()
```

Аналогно, као у претходном кораку, приказујемо сада scatter плотове са расподелом аеродрома по кластерима за различите бројеве кластера, од три до десет:

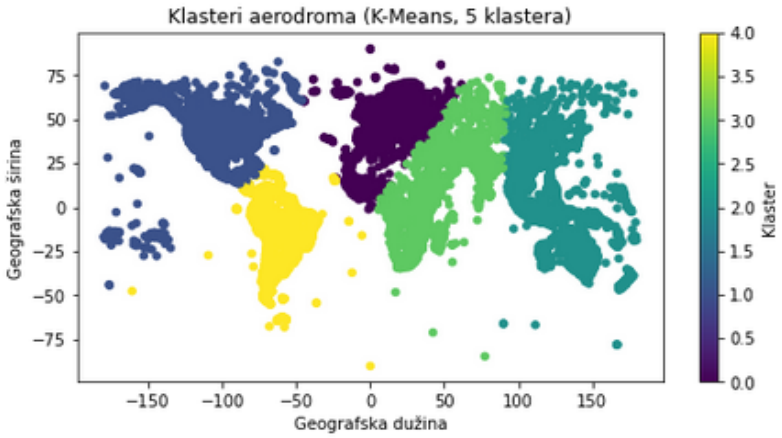
```python
for n_clusters in num_clusters:
    kmeans = KMeans(n_clusters=n_clusters, random_state=0)
    cluster_labels = kmeans.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)


    plt.figure(figsize=(8, 4))
    plt.scatter(data['Longitude'], data['Latitude'], c=cluster_labels, cmap='viridis', s=20)
    plt.xlabel('Geografska dužina')
    plt.ylabel('Geografska širina')
    plt.title(f'Klasteri aerodroma (K-Means, {n_clusters} klastera)')
    plt.colorbar(label='Klaster')
    plt.show()

    print(f"Silhouette Skor za {n_clusters} klastera: {silhouette_avg}")
```



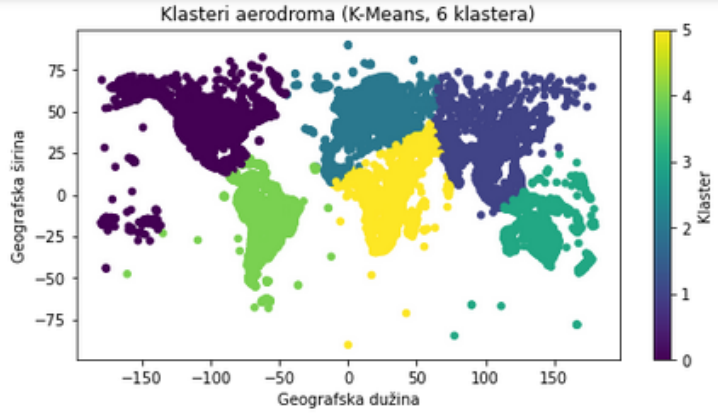

```
Silhouette Skor za 3 klastera: 0.6103316085799863
```



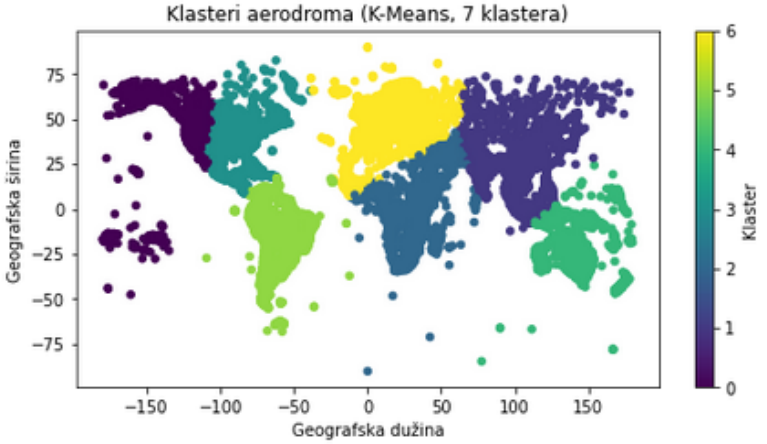

```
Silhouette Skor za 4 klastera: 0.5540660484845922
```



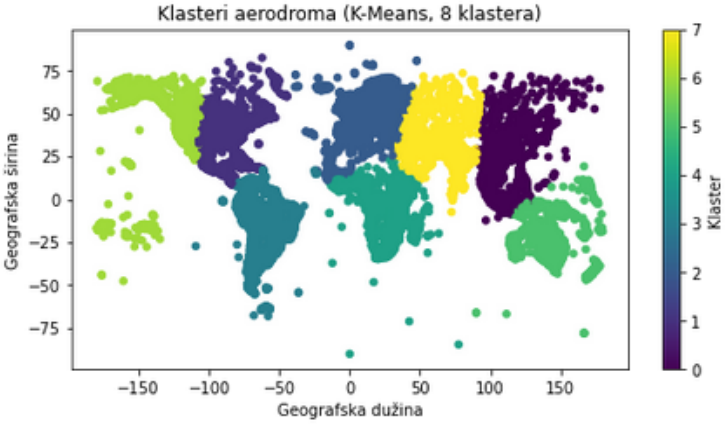

```
Silhouette Skor za 5 klastera: 0.5197414591350193
```



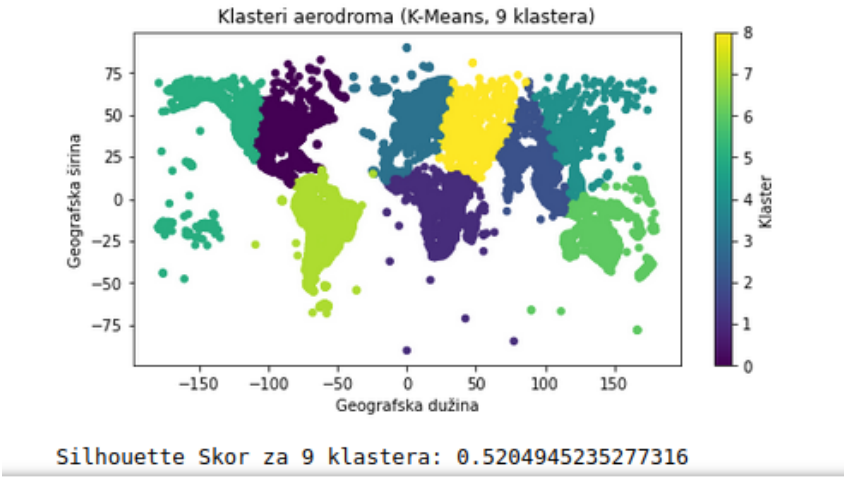

```
Silhouette Skor za 6 klastera: 0.5318882289940311
```



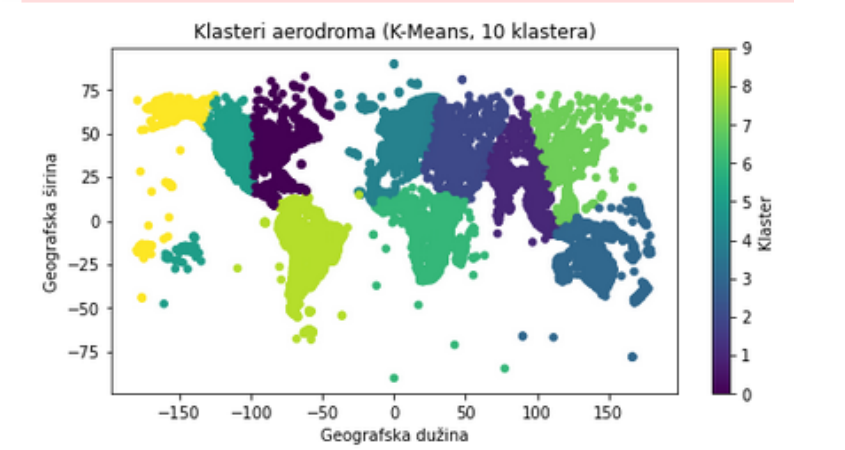

```
Silhouette Skor za 7 klastera: 0.5431095396416421
```




```
Silhouette Skor za 8 klastera: 0.5514381006449657
```




```
Silhouette Skor za 9 klastera: 0.5204945235277316
```




```
Silhouette Skor za 10 klastera: 0.5056950886949206
```

Закључак

Истраживањем података о аеродромима са openflights сајта и на основу свих резултата кластеровања, долазимо до следећих закључака:

1. **К-средина:**
    - Коришћењем овог алгоритма постигли смо доста добре резултате

- Кластери су глобуларни, што значи да су добро апроксимирани аеродроми који су близу једни другима по географској локацији.
- Овај алгоритам се показао као отпоран на присуство екстремних вредности у подацима, али мана је што број кластера мора бити унапред познат па рад захтева дужу анализу
- Резултати кластеровања су укључивали додатни транспорт.

2. **Хијерархијско кластеровање:**
   - Постигли смо врло сличне резултате, али са нешто нижим вредностима кофицијента силуете за већи број кластера
   - Може бити спор када користимо велики скупови података
3. **DBSCAN:**
   - Коефицијент силуете није био задовољавајући, јер DBSCAN боље функционише са подацима сличне густине и мало елемената ван граница
   - Овај алгоритам може бити користан за откривање компоненти повезаности, зато смо га и применили радије на скупу где смо аеродорме повезали на основу повезаности летова, али база са којом смо радили није имала довољно информација о истим
4. **Значај повезаности летова и додатних транспортних могућности:**
   - Због релативно малог броја међусобно повезаних аеродрома, повезаност летова није значајно променила резултате географске класификације.

- Такође, ако се прошири база података о додатном транспорту, ови атрибути могу променити резултате кластеровања

Кластеровање аеродрома на основу географске локације, повезаности летовима и узимајући у обзир додатне транспортне могућности је комплексан процес који за конкретне резултате захтева анализу још додатних различитих фактора као што су доступност саобраћајних путева, постојећи транспортни капацитети, потребе путника и терета, економска одрживост и други.

Важно је разумети да избор алгоритма кластеровања зависи од природе података и циљева анализе. У нашем случају, К-средина се показала као најбоља опција за класификацију аеродрома на основу географске локације, где смо притом узели у обзие додатне транспортне могућности, док се DBSCAN може користити за друге анализе, као што је откривање компоненти повезаности. Такође, важно је пратити раст и ажурирање базе података, јер додавање информација о летовима може променити динамику резултата.

Додатак

Примена DBSCAN алгоритма за проналазак најближе луке или железничке станице

Ово може бити корисно, на пример, за планирање овог типа транспортних рута или идентификацију главних транспортних чворова у одређеној регији. Као што смо навели у закључку примена DBSCAN алгоритма се види у овом контексту.

Имплементација идеје решења

Као и до сада имортујемо све потребне библиотеке.

Учитавамо податке о аеродромима из датотеке 'airports-extended.dat' и селектујемо само оне чворове који су типа 'port' (лука) и 'station' (железничка станица). Ови чворови ће бити коришћени за кластеровање.

Спајамо податке о лукама и железничким станицама у један скуп "transports", како бисмо га користили за кластеровање.

```
import pandas as pd
import numpy as np
from sklearn.cluster import DBSCAN
from geopy.distance import great_circle
```

```
# Ucitavanja podataka
airports = pd.read_csv('airports-extended.dat', header=None, names=['id', 'name', 'city', 'country', 'code', 'icao',
ports = airports[airports['type'] == 'port'][['id', 'latitude', 'longitude']]
railways = airports[airports['type'] == 'station'][['id', 'latitude', 'longitude']]
```

```
# Spajanje transporta
transports = pd.concat([ports, railways])
```

```
transports
```

	id	latitude	longitude
6074	7507	22.197075	113.558911
6438	7877	59.323300	18.081000
6439	7878	56.041900	12.691200
6444	7883	60.163056	24.969167
6565	8004	35.539000	133.264000
...	...	...	...
10288	11731	42.135561	24.742210
10289	11732	39.515149	116.707481
10290	11733	28.146282	113.063707
10291	11734	30.607485	114.424268
10298	11741	55.679333	12.585167

1433 rows × 3 columns

Дефинишемо епсилон (ε) који представља максимално растојање између тачака у истом кластеру. У нашем случају, епсилон је подешен на 10 километара у радијанима.

Креирамо DBSCAN модел користећи библиотеку Scikit-learn. Параметри модела укључују епсилон, минимални број тачака (min_samples) и метрику растојања (у овом случају, 'haversine' за географско растојање).

Дефинишемо функцију get_center_point која за сваки кластер рачуна средње вредности географских координата свих тачака у том кластеру. Ови центри ће бити коришћени за идентификацију географске средине кластера.

Примењујетмо функцију get_center_point на сваки кластер како бисте добили центар кластера.

```
# Definisanje epsilona
epsilon = 10 / 6371.01  # 10km u radijanima
```

```
# Definisanje DBSCAN modela
db = DBSCAN(eps=epsilon, min_samples=1, algorithm='ball_tree', metric='haversine').fit(np.radians(transports[['latitu
```

```
# Dodeljivanje klastera
transports['cluster'] = db.labels_
```

```
# Funkcija za dobijanje centra svakog klastera
def get_center_point(cluster):
    cluster_points = transports[transports['cluster'] == cluster][['latitude', 'longitude']]
    center_point = cluster_points.mean()
    return center_point.values.tolist()
```

```
# Primena funkcije na svaki klaster
transports['center_point'] = transports['cluster'].apply(get_center_point)
```

```
transports
```

	id	latitude	longitude	cluster	center_point
6074	7507	22.197075	113.558911	0	[22.195402666666666, 113.56570584999999]
6438	7877	59.323300	18.081000	1	[59.32665, 18.069528]
6439	7878	56.041900	12.691200	2	[56.029109775, 12.63790715]
6444	7883	60.163056	24.969167	3	[60.31737880952382, 25.028755619047622]
6565	8004	35.539000	133.264000	4	[35.539, 133.264]
...	...	...	...	...	...
10288	11731	42.135561	24.742210	826	[42.1355613, 24.74221]
10289	11732	39.515149	116.707481	611	[39.5119245, 116.70524050000003]
10290	11733	28.146282	113.063707	827	[28.1462817, 113.0637074]
10291	11734	30.607485	114.424268	828	[30.607485, 114.42426799999998]
10298	11741	55.679333	12.585167	664	[55.70137033333333, 12.574154444444444]

1433 rows × 5 columns

Дефинишемо функцију get_closest_airport која за сваки центар кластера рачуна растојање од свих аеродрома и проналази најближи аеродром користећи библиотеку geopy.distance.

Применимо функцију get_closest_airport за сваки центар кластера како бисте добили име најближег транспорта.

```python
def get_closest_airport(transport):
    transport_point = transport['center_point']
    airports['distance'] = airports.apply(lambda row: great_circle((row['latitude'], row['longitude']), transport_poi
    closest_airport = airports.loc[airports['distance'].idxmin()]['name']
    return closest_airport
```

```python
# Primena funkcije kako bismo dobili ime najblizeg aerodoroma od svakog centra klastera
transports['closest_airport'] = transports.apply(get_closest_airport, axis=1)

print(transports[['id', 'cluster', 'closest_airport']])
```

```
          id  cluster                    closest_airport
6074    7507        0  Macau Outer Harbour Ferry Terminal
6438    7877        1                         T-Centralen
6439    7878        2                  Helsingor Station
6444    7883        3           Koivukyla Railway Station
6565    8004        4                    Sakaiminato Port
...      ...      ...                                 ...
10288  11731      826           Plovdiv South Bus Station
10289  11732      611            Langfang Railway Station
10290  11733      827      Changsha South Railway Station
10291  11734      828               Wuhan Railway Station
10298  11741      664            Nordhavn Railway Station

[1433 rows x 3 columns]
```

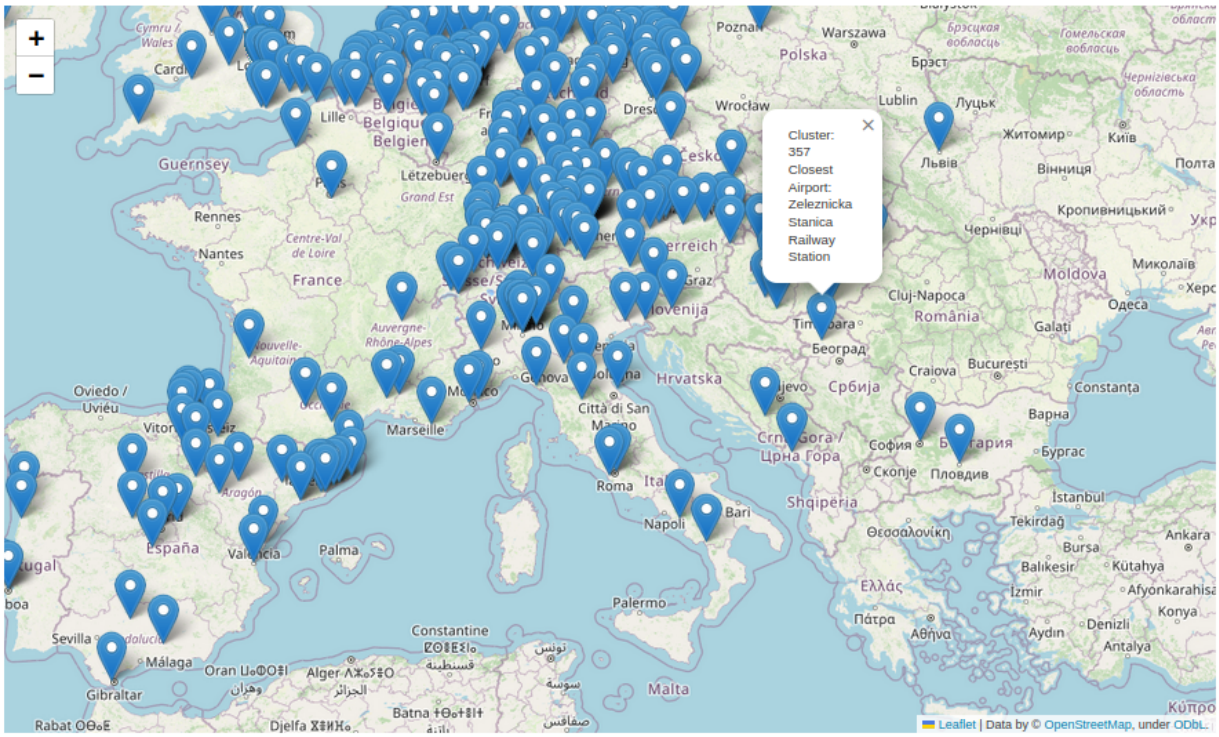
За визуелизацију користимо библиотеку folium за креирање интерактивне мапе са маркерима који представљају кластер транспортних чворова.

Креирамо мапу (м) са центром у Европи (координате Лондона) и одређеним нивоом зумирања.
За сваку централну тачку транспорта (центар кластера) креирамо маркер на мапи.

Позивањем save методе за мапу (м), она се чува као HTML датотека под називом "интерактивна_мапа.html" у нашем радном директоријуму.

Када корисник изврши овај код и отвори "интерактивна_мапа.html" у свом интернет прегледачу, може кликнути на маркере да би видео додатне информације које смо додали у искачујућим прозорима.

```python
import folium

# Kreiranje mape sa centrom u Evropi (koordinate Londona)
m = folium.Map(location=[51.5074, 0.1278], zoom_start=4)

# Kreiranje markera za svaku centralnu tacku transporta i dodavanje mapi
for index, row in transports.iterrows():
    cluster = row['cluster']
    center_point = row['center_point']
    closest_airport = row['closest_airport']
    popup_text = f"Cluster: {cluster}<br>Closest Airport: {closest_airport}"
    folium.Marker(location=center_point, popup=popup_text).add_to(m)

# Prikazivanje mape
m.save('interaktivna_mapa.html')
m
```



Покретање пројекта

Код је писан у програмском језику python, у окружењу jupyter notebook, па је погодно покренути га у истом окружењу, али пре тога инсталирати све потребне библиотеке коришћене у пројекту. (наведено пре сваког дела у склопу препроцесирања).

Заједно са овим документом биће приложени сви коришћени .ipynb фајлови.

Литература

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.calinski_harabasz_score.html
https://docs.jupyter.org/en/latest/start/index.html
https://openflights.org/data.html