AZƏRBAYCAN RESPUBLİKASI ELM VƏ TƏHSİL NAZİRLİYİ

AZƏRBAYCAN TEXNİKİ UNİVERSİTETİ

Əlyazması hüququnda

CAVİD YAŞAR OĞLU ABBASLI

K-MEDOİD ALQORİTMİ, ONUN MODİFİKASİYALARININ TƏDQİQİ VƏ TƏTBİQLƏRİ

mövzusunda

MAGİSTRLİK DİSSERTASİYASI

İxtisas: 060631 – "Kompüter mühəndisliyi"

İxtisaslaşma: "Biliklərin əldə edilməsi sistemləri"

Elmi rəhbər: **AMEA-nın müxbir üzvü, t.e.d. ,**
Ramiz Məhəmməd oğlu Alıquliyev

BAKI – 2023

MÜNDƏRİCAT

GİRİŞ 3

I FƏSIL . VERİLƏNLƏRİN İNTELLEKTUAL ANALİZİ 5

1.1. Biliklərin əldə olunması prosesi və təlim üsulları 5

1.2. Klasterləşdirmə üsulları və onların təsnifatı 9

1.3. Klasterləşdirmə üsullarının tətbiq sahələri 11

1.4. Klasterləşdirmə probleminin ümumi qoyuluşu 12

II FƏSIL . K-MEANS, K-MEDOİDS, CLARA VƏ CLARANS ALQORİTMLƏRİNİN ANALİZİ 18

2.1. K-means alqoritminin analizi 18

2.2. K-medoids alqoritminin analizi 24

2.3. CLARA alqoritminin analizi 28

2.4. CLARANS alqoritminin analizi 30

III FƏSIL . K-MEANS, K-MEDOİDS, CLARA VƏ CLARANS ALQORİTMLƏRİNİN PYTHON MÜHİTİNDƏ EKSPERİMENTAL MÜQAYİSƏSİ 33

3.1. Python mühitinin qurulması və əsas komponentlər üsulu 33

3.2. K-means alqoritminin Python mühitində aparılmış eksperimenti 38

3.3. K-medoids alqoritminin Python mühitində aparılmış eksperimenti 45

3.4. CLARA alqoritminin Python mühitində aparılmış eksperimenti 50

3.5. CLARANS alqoritminin Python mühitində aparılmış eksperimenti 55

3.6. Eksperimentlərin nəticələrinin analizi 61

NƏTİCƏ 63

İSTİFADƏ EDİLMİŞ ƏDƏBİYYAT 64

ƏLAVƏLƏR 66

GİRİŞ

Mövzunun aktuallığı. İnformasiya texnologiyalarının sürətli inkişafı nəticəsində ənənəvi üsullarla analiz oluna bilinməyən böyük həcmli verilənlərin analizi üçün bir neçə müasir texnologiyalar mövcud olmuşdur. Verilənlərin analizində faydalı biliklərin əldə edilməsi üçün istifadə olunan ən müasir texnologiyalardan biri də "Data mining" texnologiyasıdır. Verilənlərin analizi üçün "Data mining" texnologiyasında istifadə olunan ən effektiv üsullardan sayılan klasterləşdirmə üsulları müasir dünyada bir çox sahələrə tətbiq olunur.

Vendorlar və bir çox korporativ şirkətlər dillerlərinin, rəsmi nümayəndəliklərinin müxtəlif ölkələrdə yerləşdirilməsi zamanı əhalinin sayının, yüksək əhali sıxlığının və müraciətlərin çox olduğu əraziləri seçir. Praktikada bu kimi optimal seçimlər zamanı, klasterləşdirmə alqoritmlərindən istifadə olunur. Klasterləşdirmə üsulları arasında effektivliyinə görə geniş istifadə olunan alqoritmlərdən biri də K-medoids alqoritmidir.

İşin məqsədi. Dissertasiya işinin məqsədi K-means, K-medoids və onun iki fərqli modifikasiyasının – CLARA və CLARANS alqoritmlərinin tədqiqi, tətbiq sahələrinin araşdırılması və onların eksperimental müqayisəsidir.

Tədqiqat obyekti. Dissertasiya işində aparılmış eksperimentlər zamanı, tədqiqatın obyekti əhalinin demoqrafik göstəricilərini nəzərə almaqla, telekommunikasiya baza stansiyalarının optimal yerləşdirilməsidir.

İşin təcrübi əhəmiyyəti. Dissertasiya işində tədqiq olunmuş alqoritmlər praktiki əhəmiyyətli telekommunikasiya baza stansiyalarının optimal yerləşdirilməsi məsələsini həll etmişdir.

İşin aprobasiyası. Azərbaycan Dövlət Neft və Sənaye Universitetində keçirilmiş Ümummilli lider Heydər Əliyevin anadan olmasının 100 illik yubileyinə həsr olunmuş Doktorantların və Gənc Tədqiqatçıların Elmi Konfransında "Avtomatika və informatika" bölməsi üzrə "K-means alqoritminin təhlili və onun tətbiqi" adlı tezis ilə çıxış edilmişdir. Bakı 2023.

İşin strukturu və həcmi. Dissertasiya işi girişdən, 3 fəsildən, nəticədən, əlavələrdən 21 sayda ədəbiyyat siyahısından ibarətdir. İşin əsas həcmi 65 səhifə mətndən, 12 şəkildən, 5 cədvəldən, 2 qrafikdən ibarətdir. İşin ümumi həcmi 77 səhifədir.

I FƏSIL . VERİLƏNLƏRİN İNTELLEKTUAL ANALİZİ

1.1. Biliklərin əldə olunması prosesi və təlim üsulları

Verilənlərin intellektual analizi keçən əsrin 70-80-ci illərindən başlayaraq böyük həcmli verilənlər bazasından, verilənlər anbarından və verilənlər gölündən faydalı biliklərin əldə edilməsi prosesinə önəmli dərəcədə töhvə vermişdir. KDD *(eng: Knowledge Discovery in DataBases)* – yəni, "verilənlər bazasından biliyin çıxarılması" prosesinin analiz mərhələsi olan "Data Mining" mərhələsi, verilənlərin intellektual idarə olunması, ilkin emalı, onların modelləşdirilməsi yanaşmaları və ilkin emal sonrası verilənlərin strukturunun təyini, vizuallaşdırılması kimi bir çox məsələlərlə məşğul olur. Haqqında danışdığımız KDD prosesini daha aydın anlamağımız üçün bu prosesin mərhələlərinə baxaq:

1) **Verilənlərin təmizlənməsi** – verilənlər bazasındakı "küy" tipli verilənləri və qeyri-konservativ olan verilənləri bazadan kənarlaşdırma mərhələsidir.
2) **Verilənlərin inteqrasiyası** – müxtəlif verilənlər bazasından seçilmiş verilənlərin tamlığını, dəqiqliyini və aktuallığını qorumaq üçün xarakteristik verilənlərin eyni verilənlər bazasında saxlanılması, inteqrasiya olunması mərhələsidir.
3) **Verilənlərin seçilməsi (və ya azaldılması)** – verilənlər bazasından məqsədyönlü etalon biliyin xarakteristikalarına əsasən əlaqəli verilənlərin seçilməsi mərhələsidir.
4) **Verilənlərin transformasiyası** – müxtəlif verilənlər bazasından seçilmiş verilənlərin modellərinin, strukturunun, qiymətinin dəyişdirilməsi mərhələsidir.
5) **Data mining** - faydalı biliklərin çıxarılması üçün intellektual metodların tətbiq edildiyi analiz mərhələsidir.
6) **Biliklərin təsvir olunması** – bu mərhələdə aşkarlanmış bilikləri istifadəçilərə təqdim etmək üçün vizuallaşdırma və biliklərin təqdim edilməsi üsullarından istifadə olunur.

Qeyd edək ki, müasir dövrdə verilənlərin təmizlənməsi, inteqrasiyası, seçilməsi və transformasiyası prosesləri ümumiləşdirilərək "xam" verilənlərin ilkin emalı mərhələsi *(eng: Pre-processing step of raw data)* olaraq da istifadə olunur.

Verilənlərin intelektual analizi və onun üsulları kifayət qədər dərin bir elm sahəsi olsa da, bu elm özü də daha geniş bir elmin, verilənlər elminin *(eng: Data Science)* ən müasir və inkişaf edən sahəsi sayılır. Verilənlər elminin əsasını birbaşa olaraq verilənlər, onlar üzərində istifadə olunan müxtəlif analizlər və üsullar təşkil edir. Verilənlərin ilkin emalı intellektual analiz, o cümlədən, verilənlər bazasından biliyin çıxarılması prosesi üçün önəmli mərhələ sayılır.

Verilənlərin ilkin emalı mərhələsində olunan analizlər "Data mining" üsulları üçün önəmli mərhələ sayılır. Bu mərhələdə verilənlər xüsusi və məhdud sayda sinifləndirilir. Sinifləndirilmiş verilənlər *(eng: Labeled Data)* intellektual analiz mərhələsi üçün səmərəli olur və "oxunaqlı" hala düşür. İlkin emal mərhələsində bütün verilənlər sinifləndirilməyə bilər. Bu halda sinifləndirilməmiş verilənlər küy tipli və ya əhəmiyyətsiz verilənlər *(eng: Noise or Outlier Data)* kimi adlandırılır.

Verilənlər elmini təhlil edərkən bir neçə əsas anlayışlarla tanış olmalıyıq. Belə ki, verilənlər əsasən atributlarla təsvir olunurlar. Atribut anlayışı verilənlərin xarakteristikalarını və xüsusiyyətlərini təmsil edir. Bu anlayış, verilənlər anbarında *"ölçü" (eng: Data dimension)* kimi, maşın öyrənməsində isə *"əlamət" (eng: Data feature)* kimi adlandırılır. Verilənləri təsvir etmək üçün istifadə olunan atributlar çoxluğuna isə əlamət vektoru *(eng: Feature vector)* deyilir. Atributların növləri verilənlərin mümkün qiymətlərinə əsasən təsnif olunur. Bu təsnifat, adlandırmaya, sıralamaya – ranka əsasən və ya ədədi və binar ola bilər.

Verilənlər üçün təyin edilmiş atributların aşağıdakı təsnifatına baxaq [9]:

1) **Nominal atributlar:** Bu təsnifatlandırma adlandırmaya əsasən gerçəkləşdirilir. Nominal atributların qiymətləri verilənlərin adlarını təmsil edən simvollardır. Hər bir qiymət müəyyən kateqoriyanı, kodu və ya bir fikri ifadə edə bilər. Nümunə üçün fərz edək ki, əhalinin xarakteristikalarını

özündə saxlayan müəyyən verilənlər bazası mövcuddur. Məlumdur ki, bu verilənlər bazası üçün *sosial_durum, yaş_aralığı* və *təhsil_həyatı* adlı dəyişənlər verilənlər bazasına daxil edilə bilər. Onda, bu dəyişənlər üçün *işsiz, yetkin* və *orta_təhsil* kimi müxtəlif qiymətləri mənimsədə bilərik. Bu halda məlum dəyişənlər nominal atributlar adlanır.

2) **Binar atributlar:** Binar atributlaşdırma nominal atributlaşdırmanın bir növü olub, yalnızca iki qiyməti özündə saxlayır. Bu qiymətlər, 0 – *False* və 1 – *True* olaraq adlanır. Bul cəbri kursundan da məlum olduğu kimi cari verilən məlum atributlaşdırmaya uyğun gəldiyi halda verilən üçün *1* qiyməti, uyğun gəlmədiyi halda isə *0* qiyməti təyin olunur.
3) **Ordinal atributlar:** Bu atributlar müəyyən nizamlanmaya, bütün mümkün və artan(və ya azalan) sıralamaya, ranka uyğun qiymətləri özündə saxlayır. Fərz edək ki, *tələbənin_qiyməti* adlı nominal atribut üçün aşağıdakı qiymətlər təyin olunmuşdur: *əla, çox_yaxşı, yaxşı, kafi*. Məlumdur ki, qiymətlər müəyyən azalan tendensiyaya malik olaraq sıralanmışdır. Bu halda məlum nominal atribut ordinal atribut olaraq adlandırılır. Digər bir nümənədə isə insanın üz cizgilərinə əsasən emosiyaların təyinini misal göstərə bilərik:

var emosiya_numune = {qemgin, aghlayan, sakit, guleruz, xoshbext}

4) **Ədədi atributlar:** Verilənlər üzərində kəmiyyət bildirən atributları təyin etmək üçün istifadə edirik. Ədədi atributlaşdırma praktikada geniş istifadə olunan atributlaşdırma növü olub, verilənlər üçün ədədi qiymətlər təyin edir.

KDD prosesini təhlil edərkən, bu prosesdə biliklərin çıxarılması üçün intellektual metodların tətbiq edildiyi analiz mərhələsi olan “Data mining” texnologiyasının sistematik analizi zamanı aydın olur ki, proses riyazi statistika, maşın öyrənməsi *(eng: Machine Learning)*, proqramlaşdırma, verilənlərin vizuallaşdırılması və alqoritmləşdirilmə kimi istiqamətləri özündə ehtiva edir. Statistik modellərin qurulması verilənlərin riyazi təsvir olunaraq onun oxunaqlılığını təmin edirsə, maşın öyrənməsi isə birbaşa olaraq “Data Mining” üsullarını tətbiq edərək, kompüterlərin verilənlər əsasında necə öyrənə biləcəyini - performansını hansı üsullarla yaxşılaşdıra

biləcəyini araşdırır. Maşın öyrənməsinin əsas tədqiqat sahəsi kompüter proqramlarının mürəkkəb nümunələri tanıması və verilənlər əsasında ağıllı qərarlar qəbul etməsi prosesini avtomatlaşdırmaqdır.

Maşın öyrənməsinin bir neçə növü mövcuddur:

1. **Təlimli öyrənmə** – *(eng: Supervised Learning)* xam verilənlər ilkin emal prosesindən sonra təlim görmüş nümunələr və modellər üzrə sinifləndirilir. Təlimli öyrənmədə klassifikasiya (və ya təsnifatlandırma) və reqressiya üsullarından istifadə olunur. Neyron şəbəkələr və xətti reqressiya modelləri, Naive-Bayes klassifikasiyası təlimli öyrənmənin əsaslarından sayılır.
2. **Təlimsiz öyrənmə** - *(eng: Unsupervised Learning)* cari təlim görmüş nümunələr üzrə sinifləndirilə bilinməyən xam verilənlər üçün yeni modellər yaradılır və ilkin emal prosesindən sonra yeni təlim modelləri üzrə təsnif olunur. Təlimsiz öyrənmədə bölünməyə əsaslanan, ierarxik klasterizasiya və assosiasiya üsullarından istifadə olunur.
3. **Yarım təlimli öyrənmə** - *(eng: Semi - supervised Learning)* ilkin emal prosesindən sonra bəzi verilənlər cari təlim görmüş nümunələr üzrə sinifləndirilə bilsə də, bəzi verilənlər cari təlim görmüş nümunələr üzrə sinifləndirilə bilinmir. Bu halda həm təlimli öyrənmə üsullarından, həm də təlimsiz öyrənmə üsullarından istifadə olunur.
4. **Gücləndirmə ilə öyrənmə** - *(eng: Reinforcement learning)* mühitlə interaktiv olaraq əks-əlaqə sisteminə əsaslanan təlimli öyrənmə üsuludur. Qurulmuş model problemin həlli zamanı tapdığı hər uğurlu həll yolu üçün mükafatlandırılır. Əks halda isə, cəzalandırılır. Bu "cəza-mükafat" prinsipi adlandırılır və modelin optimal həll yolunu tapması üçün bir istiqamət verir. Bu öyrənmə üsulunun aktiv və passiv öyrənmə növləri mövcuddur. Aktiv öyrənmə, ətraf mühitin mövcud vəziyyətinə əsaslanaraq görüləcək tədbirləri aktiv şəkildə seçən agentdir. Bu o deməkdir ki, agent öz hərəkətləri üzərində tam nəzarətə malikdir və mükafatlarını artırmaq üçün ən yaxşı yolu müəyyən etmək üçün müxtəlif variantları araşdırmaqda sərbəstdir. Passiv öyrənmədə, həll üsulları əvvəlcədən

proqramlaşdırılmış alqoritm xarici agent tərəfindən müəyyən edilir. Belə hallarda agent yalnız öz hərəkətlərinə görə mükafat və ya cəzalar alır və məqsədi öz fəaliyyətini rəylərə əsasən yaxşılaşdırmaqdır.

1.2. Klasterləşdirmə üsulları və onların təsnifatı

Klaster analizi verilənlərin müəyyən sayda klasterlərdə yerləşdirilməsi – klasterləşdirilməsi üsullarını öyrənir. Verilənlərin bu cür klasterlərdə yerləşdirilməsi zamanı hər bir klasterdəki verilənlər müəyyən əlamətlərinə görə yerləşdirilir. Fərqli klasterlər bir-birindən xarakteristikalarına görə fərqlənir. Qeyd edək ki, seçilmiş verilənlər üzərində müxtəlif klasterləşdirmə alqoritmlərindən istifadə oluna bilər.

Verilənlərin klasterləşdirilməsi zamanı istifadə olunan alqoritmlər üçün müəyyən tələblər qoyula bilər. Bu tələblər klasterləşdirmə üsulunun tətbiqi üçün önəmli sayılır. Klaster analizi üçün qoyulmuş tələblər aşağıdakılardır [9]:

- **Miqyaslanabilən olması**: Verilənlərin intellektual analizi zamanı miqyaslanabilən klasterləşdirmə alqoritmlərindən istifadə olunmalıdır. Bir çox klasterləşdirmə alqoritmləri, özündə bir neçə yüz veriləni ehtiva edən verilənlər bazası üçün məhsuldar sayılır və zaman mürəkkəbliyi cəhətdən səmərəlidir.
- **Müxtəlif növ atributların tətbiq oluna bilməsi**: Verilənlərin intellektual analizi zamanı istifadə olunan alqoritmlər əksər hallarda ədədi atributlandırılmış verilənlər üzərində realizə olunsa da, klasterləşdirmə alqoritmləri binar, nominal, ordinal, həmçinin kombinə olunmuş atributlandırmadan istifadə olunmuş verilənlər üçün də uğurla tətbiq oluna bilinir.
- **İxtiyari formaya malik klasterlərin tətbiq edilə bilməsi**: Klasterlərin həndəsi forması bilik axtarışı prosesi üçün maneə yaratmır və optimal həllin tapılmasına zəmanət verir.
- **İnterpretasiya oluna bilməsi**: Klasterləşdirmə alqoritmlərinin qurulması zamanı iterasiyaların sayı öncədən təyin olunur. Həmçinin alqoritmlər üçün öncədən təyin olunmuş iterasiyaların sayı, yenidən fərqli iterasiyalarda realizə oluna bilər.

Məlumdur ki, klasterləşdirmə müəyyən əlamətləri bənzər olan verilənləri qruplaşdırmaq üsuludur. Bu üsullar, manipulyasiya oluna bilən sinifləndirilməmiş verilənlər çoxluğunda oxşar əlamətlərin aşkar olunması proseslərini özündə əks etdirir.

Klaster analizinin əsaslarından sayılan, ilkin və sadə klasterləşdirmə üsullarına və onların təsnifatına nəzər salaq [9]:

Bölünməyə əsaslanan klasterləşdirmə üsulları: Bölünməyə əsaslanan klasterizasiya üsullarının tətbiqi zamanı qurulmuş model verilmiş n sayda verilən ibarət olan $P=\{X_1, X_2, X_3 \dots X_n\}$ çoxluğu üzərində k sayda alt çoxluq yaradır. Hər bir altçoxluq bir klasteri təmsil edir və zəruri olaraq $k \leq n$ şərti qoyulur. Zəruri şərtdən aydın olur ki, hər bir klasterdə minimum bir verilən yerləşdirilməlidir. Sadə və ilkin yanaşma zamanı invariant olaraq tapılmış həll, hər bir verilənin bir klasterdə yerləşdirilməsi kimi başa düşülür. Bu halda, k - klaster sayı verilənlərin sayına bərabər olmuş olur, yəni, $n = k$ şərti ödənilir.

Bir çox bölünməyə əsaslanan klasterləşdirmə üsulları verilənlərin Hilbert müstəvisində uyğun, məxsusi koordinatlarları arasındakı məsafəyə əsaslanır. Klasterləşdirmənin realizə olunması zamanı iterativ yerdəyişmələrdən istifadə olunur. İterativ yerdəyişmələr klasterləşdirmənin keyfiyyətini təmin etmək üçün, hər iterasiyada cari klasterə adi edilmiş verilən ya yeni klasterə daşınır, ya da olduğu kimi saxlanılır. Verilənlərin optimal klasterləşdirilməsi zamanı bölünməyə əsaslanan klasterizasiya üsullarından bir neçə alqoritmlər eyni anda istifadə oluna bilər. Bölünməyə əsaslanan klasterləşdirmə alqoritmlərindən sadəliyinə və effektivliyinə görə K-means, dəqiqliyi və keyfiyyətli klasterləşdirməsinə görə isə K-medoids, CLARA, CLARANS alqoritmlərini misal göstərə bilərik.

İerarxik klasterləşdirmə üsulları: Bu üsulda verilənlərin ierarxik dekompozisiyası qurulur. İerarxik dekompozisiyanın qurulması zamanı iki növ yanaşmadan istifadə oluna bilər. Birinci yanaşma, aqlomerativ – toplayıcı *(eng: agglomerative approach)* yanaşma adlanır. Bu dekompozisiya zamanı verilənlərin hər birinə bir klaster kimi baxılır. Hər növbəti iterasiyalarda müəyyən əlamətləri bənzər

olan verilənlər eyni klasterlərə aid edilir. İkinci yanaşma isə bölünmə - parçalanma qaydası ilə yanaşmadır. Bu yanaşmada isə bütün verilənlərə bir klaster kimi baxılır və hər növbəti iterasiyada "oxşar" verilənlər fərqli klasterlərə aid edilir.

Sıxlığa əsaslanan klasterləşdirmə üsulları: Sıxlığa əsaslanan klasterləşdirmə zamanı digər üsullardan fərqli olaraq iterativ bölmələr verilənlər arasındakı məsafəyə əsasən realizə olunmur. Sıxlığa əsaslanan klasterləşdirmə üsulları verilənlərin daha sıx səpələndiyi mərkəzlər klaster mərkəzləri olaraq seçilir. Eyni anda məsafəyə və sıxlığa əsaslanan klasterləşdirmə üsullarını tətbiq etmək optimal klasterləşdirmə üçün daha uyğun sayılır.

1.3. Klasterləşdirmə üsullarının tətbiq sahələri

Praktikada bir çox əhəmiyyətli, xüsusilə biznes analitikası, ehtimal olunan gəlirlər və məzənnə dəyişimi kimi məsələlərin həlli üçün optimal seçimlərin edilməsi verilən elminin predmeti olaraq tanınır. Müasir dünyada yayılan geniş miqyaslı biznes strategiyaları və onların optimal idarə edilməsi üçün böyük həcmli verilənlər bazası ilə işləmək bacarığı tələb olunur. Yüksələn tendensiyaya malik şirkətlər, o cümlədən hər saniyədə milyonlarla satış edən və qlobal bazarda hökmranlıq edən kompaniyaların çoxu satış strategiyalarını müəyyən etmək üçün bu texnologiyalardan istifadə edirlər.

O cümlədən, klasterləşdirmə alqoritmləri praktikada bir çox böyük ölçülü məsələlərin həlli üçün tətbiq olunur. Bu cür məsələlər özündə böyük həcmli verilənləri ehtiva etdiyi üçün ənənəvi üsullarla həll oluna bilinmir. Buna görə də müasir texnologiyalardan olan "Data Mining" böyük həcmli verilənlərlə işləmək üçün ən güclü instrumentlərdəndir. Klasterləşdirmə alqoritmləri bir-biriləri ilə dəqiqliyi və yaradılmış klasterlərin keyfiyyəti ilə fərqlənsə də, tətbiq sahələri bənzərdir.

Klasterləşdirmə üsullarının **tətbiq sahələri** aşağıdakılardır:

1) Telekommunikasiya baza stansiyaların optimal yerləşdirilməsi
2) Tibb ocaqlarının, habelə apteklərin inşası üçün optimal yerin seçilməsi
3) Kompüter şəbəkələrində hub-ların seçilməsi
4) Mətnlərin məzmuna görə qruplaşdırılması

5) Elektron ticarətdə yeni trendlərin müəyyən edilməsi
6) Supermarketlərin inşası üçün optimal yerin seçilməsi
7) Dövlət sənədlərinin kateqoriyalandırılması
8) DoS hücumların aşkarlanması və s.

Bununla belə klasterləşdirmə alqoritmləri bir çox optimal yerləşdirmə məsələlərini həll edir. Yuxarıda qeyd etdiyimiz kimi müxtəlif alqoritmlərin baxılan məsələlərə tətbiqi zamanı, yaranmış klasterlərin keyfiyyəti və klaster mərkəzləri fərqli göstəricilərə malik olacaqdır. Baxılan, telekommunikasiya baza stansiyaların optimal yerləşdirilməsi məsələsində bölünməyə əsaslanan klasterləşdirmə alqoritmlərin tədqiqi zamanı bunu müşahidə edəcəyik.

1.4. Klasterləşdirmə probleminin ümumi qoyuluşu

Klasterləşdirmə problemini şərh edərkən, bu problemin minimallaşdırma məsələsi olduğunu aydın şəkildə görə bilərik. Problemin riyazi qoyuluşuna baxmadan öncə bir neçə əsas anlayışları daxil etməliyik. Belə ki, riyazi statistika kursundan məlumdur ki, ədədi sıranının orta qiymətinin, modasının və medianının tapılması zamanı müəyyən ədədi sıraya sonlu ardıcıllıq kimi baxılır. Verilənlər bazasını sonlu çoxluq və bu bazadakı verilənlərə isə çoxluğun elementləri kimi baxaraq bu anlayışları riyazi şərh edək.

Fərz edək ki, n sayda veriləndən ibarət olan sonlu $P=\{X_1, X_2, X_3 \dots X_n\}$, $\forall X_i \in R^m$, $i = \overline{1,n}$ çoxluğu m ölçülü Hilbert fəzasında təyin olunmuşdur. Hər bir X_i elementi üçün,

$$X_i = (x_{i1}, x_{i2}, x_{i3} \cdots x_{im}), \forall x_{ij} \in X_i, j = \overline{1,m} \qquad (1.1)$$

koordinatları verilmişdir. Qeyd edək ki, koordinat vektorunun elementlərindəki işarələmədə x_{ij} elementi *i* nömrəli X_i elementin *j* nömrəli R^j fəzasındakı koordinatını təmsil edir.

Həmçinin, $P=\{X_1, X_2, X_3 \dots X_n\}$ çoxluğunu aşağıdakı iki ölçülü koordinatlar matrisi kimi göstərə bilərik:

$$P = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ X_n \end{pmatrix} = \begin{Vmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1m} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2m} \\ x_{31} & x_{32} & x_{23} & \cdots & x_{3n} \\ \vdots & & \cdots & & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nm} \end{Vmatrix} = \|x_{ij}\|_{n\,x\,m} \qquad (1.2)$$

İlk öncə verilənlərin statistik orta ədədi qiymətini tapaq. Bunun üçün X^{orta} elementinin hər bir j nömrəli fəzadakı koordinatlarının orta nöqtəsini tapmalıyıq. Qeyd edək ki, statistik orta ədədi qiymətə malik verilən real verilən olmaya bilər.

Belə ki,

$$X^{orta} = \left(\frac{\sum_{i=1}^{n} x_{i1}}{m}, \frac{\sum_{i=1}^{n} x_{i2}}{m}, \frac{\sum_{i=1}^{n} x_{i3}}{m} \dots, \frac{\sum_{i=1}^{n} x_{im}}{m}\right) \qquad (1.3)$$

Aydındır ki, statistik orta qiymətin tapılması klasterləşdirmə prosesinin aparılması üçün önəmli şərtdir. Statistik ardıcıllığın orta qiymətini tapmaqla yanaşı, medoid anlayışını da daxil etməliyik.

Yuxarıda qeyd etdiyimiz kimi verilənlərin analizi zamanı statistik orta verilən əksər hallarda həqiqi veriləni təmsil etməyə bilər. Lakin, praktiki əhəmiyyətli məsələlərin həlli zamanı məlum olur ki, verilənlər üzərində klasterləşdirmə aparılan zaman klasterlərin mərkəzi seçilərkən bu verilənin həqiqi verilən olması zəruri şərtə çevrilir. Nümunə üçün müəyyən ərazilərdə yerləşdirilmiş apteklərə dərmanların optimal paylanması problemini təhlil edərkən, optimal paylanmanı klasterləşdirmə alqoritmləri üzərindən qursaq, aydın olur ki, klasterlərin mərkəzlərini seçərkən zaman kriteriyasını nəzərə alaraq mərkəzi obyektin həqiqətən də aptek olmasına əmin olmalıyıq. Əks halda

optimal həll doğru qurulmayacaqdır. Aydındır ki, baxılan məsələlərdə obyektlər və onlar yerləşdiyi metrik fəza üçün tətbiq olunmuş müəyyən metrik normalara əsasən klaster mərkəzlərinin seçilməsi mühim şərtdir.Bu səbəbdən də metrik norma anlayışı ilə tanış olaq.

Funksional analiz və analitik həndəsə kursundan məlumdur ki, m ölçülü Evklid fəzasında təyin olunmuş sonlu X çoxluğunun $\forall(x, y)$ elementlər cütü üçün bu elementlər arasında müəyyən $\rho(x, y) := X \times X \to R^m$ məsafə funksiyası qarşı qoyulursa və aşağıdakı şərtlər ödənirsə, bu inikas metrika adlanır:

1. Pozitivlik şərti: $\rho(x, y) \geq 0$ həmişə doğrudur və $\rho(x, y) = 0$ halı yalnız $x = y$ olduğu halda doğrudur.
2. Simmetriklik şərti: $\rho(x, y) = \rho(y, x)$
3. Üçbucaq bərabərsizliyi şərti: $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$

Həmçinin, metrikanın tətbiq olunduğu fəza metrik fəza adlanır.Məsələn, n ölçülü Evklid fəzasında müəyyən məsafə funksiyası – metrik norma istifadə olunduğu zaman, (R^m, ρ) işarələməsi aparılır.Bəzi ədəbiyyatlarda $\rho(x, y)$ metrik norması məsafə anlayışını *(eng: distance – abbr: dist)* özündə ehtiva etdiyindən sadəlik üçün $dist(x, y)$ kim də qeyd olunur.

Yuxarıda qeyd etdiyimiz kimi klasterləşdirmə zamanı obyektlər arasındakı məsafələrin hesablanması üçün müəyyən metrik normalardan istifadə oluna bilər. Aşağıdakı metrik normalara baxaq:

1. L_1 metrik norması – (R^m, L_1) metrik fəzasında tətbiq olunur və bu məsafə funksiyası obyektlər arasındakı Manhattan məsafəsi və ya Taksi məsafəsi olaraq adlanır. Belə ki, m ölçülü Evklid fəzasıında təyin olunmuş sonlu X çoxluğunun $\forall(x, y)$ elementlər cütü üçün metrik norma bu cür hesablanır:

$$L_1 = \rho(x, y) = \|x - y\|_1 = \sum_{i=1}^{n} |x_i - y_i| \qquad (1.4)$$

2. L_2 metrik norması – (R^m, L_2) metrik fəzasında tətbiq olunur və bu məsafə funksiyası obyektlər arasındakı Evklid məsafəsi olaraq adlanır. Belə ki, n ölçülü Evklid fəzasıında təyin olunmuş sonlu X çoxluğunun $\forall(x, y)$ elementlər cütü üçün metrik norma bu cür hesablanır:

$$L_2 = \rho(x, y) = \|x - y\|_2 = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (1.5)$$

Beləliklə, verilənlərin analizi zamanı istifadə olunan bir neçə əsas riyazi anlayışlarla tanış olduq. Klasterləşdirmə problemi ümumi şəkildə aşağıdakı kimi təyin olunur.

Fərz edək ki, n sayda veriləndən ibarət olan sonlu, qabarıq və məhdud $P=\{X_1, X_2, X_3 \dots X_n\}$, $\forall X_i \in R^m$, $i = \overline{1,n}$ çoxluğu metrik (R^m, L_2) Hilbert fəzasında təyin olunmuşdur. Məqsədimiz metrik fəzada paylanmış verilənlərin k sayda elementdən ibarət olan

$$C=\{C_1, C_2, C_3 \dots C_k\}, \forall C_l \in R^m, l = \overline{1,k} \qquad (1.6)$$

klasterlər çoxluğunun elementlərində, yəni $C_1, C_2, C_3 \dots C_k$ klasterlərində yerləşdirməkdir. Məlumdur ki,

$$\forall\, C_l \subset P \text{ və } C_1 \cup C_2 \cup C_3 \cup \dots \cup C_k = P \qquad (1.7)$$

Həmçinin,

$$C_l \cap C_m = \emptyset\,,\ m = \overline{1,k} \text{ və } k \leq n \qquad (1.8)$$

zəruri şərti verilmişdir. Xüsusi $k = n$ halında hər bir element bir klasteri təmsil edir.

Klasterlərin mərkəzləri çoxluğunu aşağıdakı kimi göstərək:

$$O = \{\theta_1, \theta_2, \theta_3 \dots \theta_k\} \text{ və } \forall\theta_l \in C_l \subset P \qquad (1.9)$$

Bu problemin optimallaşdırma məsələsinə gətirilməsi zamanı müəyyən məqsəd funksiyasından istifadə edəcəyik. Həllin varlığının axtarılması zamanı məqsəd funksiyası təyin olunur. Fərz edək ki, optimallaşdırma məsələsinin məqsəd funksiyası aşağıdakı kimi verilmişdir [15]:

$$J = \sum_{l=1}^{k} \sum_{i=1}^{n} B_{il}\, \rho(X_i\, , \theta_l\,) \;\rightarrow min \tag{1.10}$$

$$B_{il} = \begin{cases} 1\,, əgər\; X_i \in C_l \\ 0, əgər\; X_i \;\notin\; C_l \end{cases} \tag{1.11}$$

Həmçinin,

$$B_{il} = 1\; isə, l = \arg min_l \rho(X_i\, , \theta_l) \tag{1.11}$$

Optimal klaster mərkəzlərini tapmaq üçün məqsəd funksiyasının θ_l mərkəzinə əsasən xüsusi törəməsini tapaq:

$$\frac{\partial J}{\partial \theta_l} = \frac{\partial \sum_{i=1}^{n} B_{il} \rho(X_i, \theta_l\,)}{\partial \theta_l} = \frac{\partial \sum_{i=1}^{n} B_{il} \|X_i - \theta_l\|_2^2}{\partial \theta_l} = 0 \tag{1.12}$$

Verilənlər məxsusi koordinatları ilə təmsil olunur, və X_i elementinin koordinatlarının sütun vektor kimi göstərmək əvəzinə onları transponizə edirik:

$$\frac{\partial J}{\partial \theta_l} = \frac{\partial \sum_{i=1}^{n} B_{il} \left(X_i^T \theta_l - 2\, X_i^T \theta_l + \theta_l^T \theta_l\right)}{\partial \theta_l} = 0 \tag{1.13}$$

Beləliklə bir neçə sadələşdirmə işləri apararaq,

$$\frac{\partial J}{\partial \theta_l} = \sum_{i=1}^{n} B_{il}\, (-2X_i + 2\theta_l) = 0 \tag{1.14}$$

$$\frac{\partial J}{\partial \theta_l} = -2 \sum_{i=1}^{n} B_{il}\, 2X_i + 2\theta_l \sum_{i=1}^{n} B_{il} = 0 \tag{1.15}$$

aşağıdakı nəticəni ala bilərik:

$$\theta_l = \frac{\sum_{i=1}^{n} B_{il} X_i}{\sum_{i=1}^{n} B_{il}} \tag{1.16}$$

Həmçinin baxılan məqsəd funksiyasının θ_l mərkəzinə əsasən ikinci tərtib xüsusi törəməsini əsasən aşağıdakı nəticəni alarıq:

$$\frac{\partial^2 J}{\partial \theta_l^2} = 2\sum_{i=1}^{n} B_{il} \qquad (1.17)$$

θ_l mərkəzi cari C_l klasterinin mərkəzi olduğundan dolayı aydın olur ki, $B_{il} = 1$ şərti ödənir. Belə ki, bu şərti

$$2\sum_{i=1}^{n} B_{il} > 0 \qquad (1.18)$$

bərabərsizliyini doğurur.

Həmçinin, məqsəd funksiyasının ikinci tərtib xüsusi törəməsinin Hess matrisinə interpretasiyasına baxsaq,

$$\mathcal{H} = \begin{bmatrix} \frac{\partial^2 J}{\partial \theta_1^2} & \cdots & \frac{\partial^2 J}{\partial \theta_1 \partial \theta_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial \theta_k \partial \theta_k} & \cdots & \frac{\partial^2 J}{\partial \theta_k^2} \end{bmatrix} \qquad (1.19)$$

aydın olur ki, matrisin baş dioqanal elementləri müsbət həqiqi qiymətlər alır, və matris müsbət sonlu matrisdir. Bu isə minimallaşdırma məsələsinin həllinin varlığını özündə ehtiva edir:

$$\frac{\partial^2 J}{\partial \theta_l^2} = \begin{bmatrix} \frac{\partial^2 J}{\partial \theta_1^2} & \cdots & \frac{\partial^2 J}{\partial \theta_1 \partial \theta_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial \theta_k \partial \theta_k} & \cdots & \frac{\partial^2 J}{\partial \theta_k^2} \end{bmatrix} = 2\sum_{j=1}^{n} B_{il} > 0 \qquad (1.20)$$

II FƏSIL . K-MEANS, K-MEDOIDS, CLARA VƏ CLARANS ALQORİTMLƏRİNİN ANALİZİ

2.1. K-means alqoritminin analizi

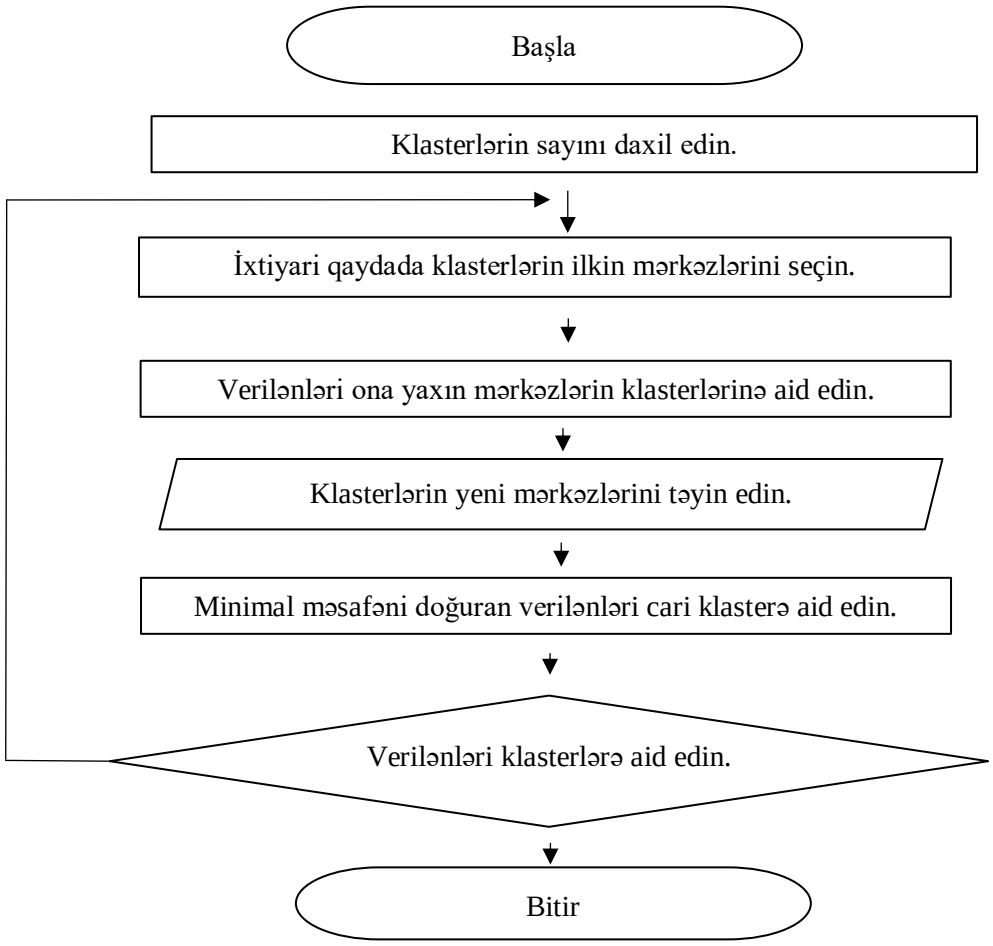
K-means alqoritmi bölünməyə əsaslanan klasterləşdirmə üsullarının əsaslarından sayılır. K-means alqoritmi *n* sayda verilənin *k* sayda klasterdə optimal yerləşdirilməsi məsələsini həll edir. Baxılan alqoritmdə, əsas məqsəd verilənlərin keyfiyyətli klasterləşdirilməsidir. Klasterləşdirmə alqoritmləri arasında ən çox istifadə olunan K-means alqoritminin tətbiq olunması asandır. Lakin, böyük həcmli verilənlərin K-means alqoritmi ilə klasterləşdirilməsi səmərəli sayılmır [8].

Alqoritmin mərhələlərini, strukturunu daha lokanik və aydın təsvir etmək üçün K-means alqoritminin addımlarını mətn şəkildə təsvir edək [21]:

1) *n* sayda veriləni klasterləşdirmək üçün klasterlərin sayını – *k* `nı təyin edin.
2) Təsadüfi yolla *k* sayda klaster mərkəzi seçin.
3) Hər bir veriləni ona məsafəcə yaxın mərkəz ətrafında qruplaşdırın.
4) Cari klasterə aid edilmiş verilənlərin statistik ədədi ortasını yeni mərkəz olaraq seçin.
5) Hər bir klasterə aid edilmiş verilənlərdən yeni təyin olunmuş mərkəzlərə qədər olan məsafələr cəmini hesablayın.
6) Klasterlərin mərkəzləri dəyişməyənə qədər (3) – (5) addımlarını təkrar edin.
7) Klasterlərin mərkəzləri dəyişmirsə, alqoritmi sonlandır.

Alqoritmin mətn şəkildə təsvirindən də göründüyü kimi verilənlərin aid edildiyi hər bir klasterin keyfiyyəti hər iterasiya üçün (5) addımında qiymətləndirilir. Belə ki, hər bir klasterə aid edilmiş verilənlərdən mərkəzə qədər olan ümumi məsafələr cəminin ədədi qiyməti nə qədər aşağı olarsa, klasterləşdirmə də bir o qədər keyfiyyətli aparılmış sayılır.

Alqoritmin sonuncu addımında görürük ki, dayanma kriteriyası birbaşa olaraq optimal həllin tapılmasını göstərir. Alqoritm son şərtli, dövri alqoritmdir və onun blok-sxemlə təsvirində də bu xüsusiyyətləri görə bilərik:



Qrafik 2.1. K-means alqoritminin blox-sxemlə təsviri

Alqoritmlərin mürəkkəblik nəzəriyyəsindən bizə məlumdur ki, zaman mürəkkəbliyi dedikdə, məsələnin ölçüsündən asılı olan elə funksiya başa düşülür ki, həmin funksiya məsələnin həllinə lazım gələn əməliyyatların maksimal sayı olsun. Bir sözlə, alqoritmin zaman mürəkkəbliyi problemin həllinə sərf olunan maksimal əməliyyat sayıdır.

Alqoritmin zaman mürəkkəbliyini analiz edərkən, alqoritmin polinomial və ya eksponensial alqoritmin olduğunu anlayırıq. Sadə dildə desək, məsələ seçilmiş alqoritmlə asan həll olunan məsələyə çevrilirsə, bu polinomial alqoritm sayılır. Əks halda isə məsələ eksponensial alqoritmlə həll olunursa, bu, çətin həll olunan məsələ sayılır. Ədəbiyyatlarda, polinomial alqoritmlərlə həll olunan məsələlər sinfi P sinfi, eksponensial alqritmlərdə isə NP sinfi olaraq təsnif olunur.

Eksponensial zaman mürəkkəbliyinə malik alqoritm üçün zəruri şərt odur ki, məsələnin həll alqoritminin zaman mürəkkəbliyini göstərən $f(n)$ funksiyası üçün $f(n) = o(g(n))$ şərtini ödəyən $\nexists P(n)$ polinomu yoxdur.

K-means alqoritmini analiz edərkən alqoritmin eksponensial alqoritm olduğunu aydın şəkildə görə bilərik. Həmçinin, K-means alqoritmi $NP - çətin$ sinifinə daxildir. Bu alqoritmin zaman mürəkkəbliyi aşağıdakı kimi təyin olunur:

$$O(n * m * k * t) \qquad (2.1)$$

Burada,

n – verilənlərin sayı

m – atributların sayı

k – yaradılan klasterlərin sayı

t – realizə olunan iterasiyaların sayı

kimi təyin olunur.

K-means alqoritmi ilə klasterləşdirmə zamanı hər növbəti iterasiyada klasterlərin mərkəzləri ilə verilənlər arasındakı məsafələr cəmi monoton olaraq azalır [17]. K-means alqoritminin realizə olunması zamanı müəyyən məhdudiyyət kriteriyaları qoyulur. Bu kriteriyalar klasterləşdirmənin keyfiyyətinə və məsələnin optimal həllinə birbaşa təsir edir.

Aşağıdakı kriteriyalara baxaq:

- **Dayanma kriteriyası**: Proses boyunca cari iterasiyadan növbəti iterasiyaya keçid zamanı, klasterlərin mərkəzləri dəyişmirsə, alqoritm sonlandırılır və optimal həll tapılır.
- **Zaman kriteriyası**: Öncədən təyin olunmuş müddət ərzində alqoritm realizə olunur. Klasterləşdirmə alqoritmləri zaman kriteriyasına əsasən bir neçə alqoritmlə eksperimental müqayisədə səmərələliliyi ilə ölçülə bilər.
- **İterasiya sayı kriteriyası**: Alqoritmin prosesə başladığı andan, öncədən təyin olunmuş və realizə olunacaq maksimal iterasiyalar sayına əsasən müəyyənləşdirilir.

Yuxarıda qeyd etdiyimiz kimi dayanma kriteriyası klasterləşdirmənin keyfiyyətli aparılması və optimal klaster mərkəzlərinin seçilməsi kimi başa düşülür. Lakin, böyük həcmli verilənlərin klasterləşdirilməsi zamanı qoyulmuş digər kriteriyalar optimal klaster mərkəzlərinin seçilməsi üçün zəmanət vermir.

K-means alqoritminin müəyyən **üstünlükləri və çatışmazlıqları** mövcuddur.Bu alqoritmin **üstünlükləri** aşağıdakılardır:

1) K-means alqoritmi ilə klasterləşdirmə digər klasterləşdirmə alqoritmlərinə nisbətən daha asandır.
2) Verilənlərin klasterləşdirilməsi zamanı K-means alqoritmi digər klasterləşdirmə alqoritmlərindən (məsələn, ierarxik klasterləşdirmə üsulları) daha qısa zaman ərzində realizə olunur.
3) K-means alqoritmi ilə klasterləşdirmə zamanı digər klasterləşdirmə alqoritmlərindən fərqli olaraq, verilənlər klaster mərkəzləri ətrafında daha sıx səpələnir.

K-means alqoritmin **çatışmazlıqları** aşağıdakılardır:

1) K-means alqoritminin ən böyük çatışmazlığı onun başlanğıc klaster mərkəzlərinin seçilməsindən asılı olmasıdır [1].

2) K-means alqoritmini böyük həcmli verilənlər üzərindən realizə edərkən, alınmış klasterlərin keyfiyyəti aşağı olur.

Qeyd edək ki, K-means alqoritmi **Lloyd alqoritmi** əsasında formalaşmışdır. Lloyd alqoritmi ilk dəfə *1957*-ci ildə *Bell Labs*-dan *Stuart P. Lloyd* tərəfindən təklif edilmişdir. Lloydun təklif etdiyi alqoritm geniş rəğbət qazansa da, *1982*-ci ilə qədər bu elmi iş nəşr olunmamışdır. Bənzər bir alqoritm müstəqil olaraq *Coel Maks* tərəfindən hazırlanmış və *1960*-cı ildə nəşr edilmişdir, buna görə də alqoritm bəzən *Lloyd-Max alqoritmi* kimi də adlandırılır.

60-cı illərdə Lloyd alqoritmində hər iterasiyadan sonra klaster mərkəzlərin yerləşdirilməsi Voronoy diaqramı üzərindən realizə olunurdu. Lloyd alqoritmi ilk dəfə Evklid fəzasında təyin olunmuş sonlu çoxluğun paylanmış elementlərinin müəyyən klasterlər daxilində qruplaşdırılması və bu klasterlərin mərkəzlərinin təyin edilməsi ilə iterativ qaydada tətbiq edilmişdir [20].

Lloyd alqoritminin aşağıdakı addımlarla realizə olunur:

1) Alqoritmi ilkin iterasiyasında iki ölçülü Evklid fəzasında təyin olunmuş k sayda elementi olan sonlu, qabarıq çoxluğun həmin elementləri təsadüfi yolla müstəvidə yerləşdirilir.
2) Daha sonra k sayda elementin yerləşdiyi müstəvi üçün Voronoy diaqramı qurulur və Voronoy bölgələrinə ayrılır. Bu bölgələr klaster anlayışının yarandığı ilkin termin olaraq sayıla bilər.
3) Yaradılmış bölgələrin (klasterlərin) ağırlıq mərkəzi və sərhədləri müəyyən olunur.

Baxmayaraq ki, bu alqoritmin və Voronoy diaqramının tətbiqi müxtəlif ölçülü fəzalar üçün də tətbiq oluna biləndir, xətanın daha aşağı olması və klasterləşdirmənin keyfiiyətli aparılması üçün bu proses əsasən (R^2, L_2) metrik fəzasında, yəni metrik Evklid fəzasında və L_2 metrik norması ilə qurulur. Voronoy çoxluqları qabarıq, sonlu və məhdud olduğundan bu klasterlər üzərində elementlərin trivial dekompozisiyası səmərəli hesab olunur. Klasterlərin həndəsi forması üçbucaq, trapesvari və dördbucaqlı

olduğu üçün bu bölgələrin ağırlıq mərkəzlərinin hesablanması zamanı analitik həndəsə kursundan məlum olan karteziyan koordinatlar üsulundan istifadə olunur.

K-means alqoritmini analitik təsviri zamanı, alqoritmin riyazi şərhinə baxacağıq. Ötən fəsildə, klasterləşdirmə probleminin optimallaşdırma məsələsinə gətirilməsi zamanı bir çox əsas anlayışlardan istifadə etmişdik. Beləliklə, **K-means alqoritminin analitik təsvirinə** baxaq:

Fərz edək ki, n sayda verilənlən ibarət olan sonlu, qabarıq və məhdud $P=\{X_1, X_2, X_3 \dots X_n\}$, $\forall X_i \in R^m$, $i = \overline{1,n}$ çoxluğu metrik (R^m, L_2) Hilbert fəzasında təyin olunmuşdur. Məqsədimiz metrik fəzada paylanmış verilənlərin k sayda elementdən ibarət olan

$$C = \{C_1, C_2, C_3 \dots C_k\}, \forall C_l \in R^m, l = \overline{1,k} \qquad (2.2)$$

klasterlər çoxluğunun elementlərində, yəni $C_1, C_2, C_3 \dots C_k$ klasterlərində qruplaşdırmaqdır.

Məlumdur ki,

$$\forall\, C_l \subset P \text{ və } C_1 \cup C_2 \cup C_3 \cup \dots \cup C_k = P \qquad (2.3)$$

Həmçinin,

$$C_l \cap C_m = \emptyset\,,\ m = \overline{1,k} \text{ və } k \leq n \qquad (2.4)$$

zəruri şərti verilmişdir. Xüsusi $k = n$ halında hər bir element bir klasteri təmsil edir.

Klasterlərin mərkəzləri çoxluğunu aşağıdakı kimi göstərsək,

$$0 = \{\theta_1, \theta_2, \theta_3 \dots \theta_k\} \text{ və } \forall \theta_l \in C_l \subset P \qquad (2.5)$$

məlum olar.

Alqoritm aşağıdakı optimallaşdırma məsələsini həll edir [12]:

$$\min_{\cup_{l=1}^{k} C_l = P} \sum_{l=1}^{k} \sum_{i=1}^{n} \|X_i - \theta_l\|^2 = \min_{\cup_{l=1}^{k} C_l = P} \sum_{l=1}^{k} \sum_{i=1}^{n} \rho(X_i, \theta_l)^2 \qquad (2.6)$$

Burada θ_l elementi C_l klasterinin mərkəzini təmsil edir və P çoxluğunun həqiqi elementidir. Fərz edək ki, X^* verilənləri C_l klasterinə aid edilmiş verilənlərdir. Onda,

$$\theta_l = \frac{1}{|C_l|}\sum_{X^* \in C_l} X^* \qquad (2.7)$$

aydın olur.

2.2. K-medoids alqoritminin analizi

K-means alqoritminin analitik təsvirindən aydın olduğu kimi klasterlərin mərkəzləri, əvvəlki iterasiyada həmin klasterlərə aid edilmiş verilənlərin statistik ədədi ortası kimi seçilir. Məlumdur ki, bu mərkəzlər heç də həmişə baxılan həqiqi verilənlər olmaya bilər. Klasterlərin mərkəzlərini təmsil edən verilənlərin həqiqiliyi üçün digər klasterləşdirmə alqoritmlərindən istifadə olunur.

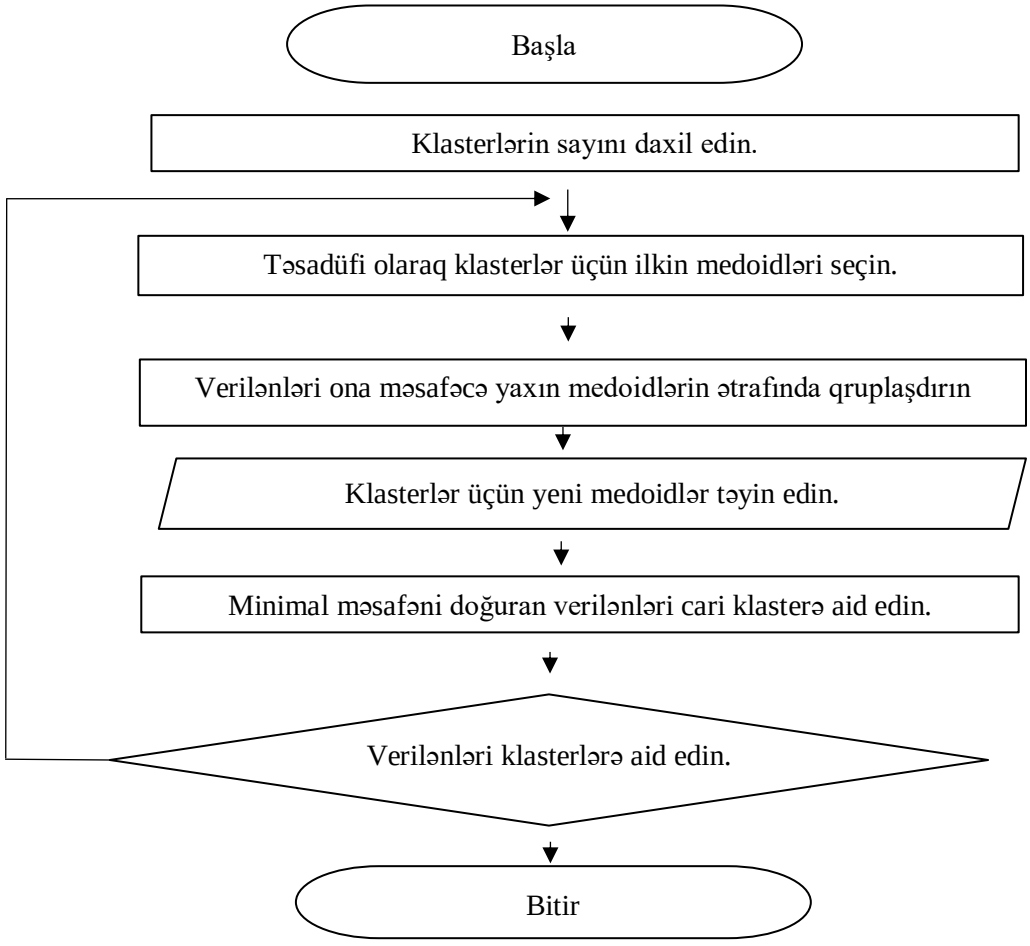
Bu alqoritmlərdən biri də birbaşa olaraq medoid anlayışı ilə əlaqəli olan seçilmiş medoidlər ətrafında bölünməyə əsaslanan üsullardır. Medoidlər ətrafında bölünməyə əsaslanan üsullar ingilis ədəbiyyatlarında *PAM (eng: Partitioning Around Medoids)* olaraq da adlandırılır. Bu alqoritmin digər adı isə **K-medoids** alqoritmidir [7].

K-medoids alqoritmi, başqa adla, medoidlər ətrafında bölünməyə əsaslanan klasterizasiya üsulu *1987*-ci ildə *Leonard Kaufman* və *Peter J. Rousseeuw* tərəfindən təklif olunmuşdur [16]. K-medoids alqoritmi K-means alqoritmində olduğu kimi *n* sayda verilənin *k* sayda klasterdə optimal yerləşdirilməsi məsələsini həll edir. Lakin, K-means alqoritmindən fərqli olaraq, K-medoids alqoritmi ilə klasterləşdirmə zamanı hər bir klasterin mərkəzi, *medoidlər* olaraq seçilir [18]. Klasterin mərkəzini təmsil edən medoid elə bir həqiqi veriləndir ki, seçilmiş medoid ilə baxılan klasterə aid edilmiş verilənlər arasındakı məsafələr cəmi minimum qiymət alsın.

K-medoids alqoritminin addımlarının daha lokanik şərhi üçün bu alqoritmi mətn şəkildə təsvir edək [16]:

1) *n* sayda veriləni klasterləşdirmək üçün klasterlərin sayını – *k* `nı təyin edin.
2) Təsadüfi yolla *k* sayda medoid seçin.
3) Hər bir veriləni ona məsafəcə yaxın medoid ətrafında qruplaşdırın.
4) Hər bir seçilmiş medoid, digər verilənlərlə əvəz olunur və hər bir əvəzetmə zamanı klasterləşdirmənin keyfiyyəti ölçülür.
5) Ən keyfiyyətli klasterləşdirmə ilə nəticələnən əvəzetmə realizə olunur və uyğun verilənlər medoid olaraq seçilir.

Alqoritm son şərtli, dövri alqoritmdir və onun blok-sxemlə təsvirində də bu xüsusiyyətləri görə bilərik:



Qrafik 2.1. K-medoids alqoritminin blox-sxemlə təsviri

Medoidlər ətrafında bölünməyə əsaslanan alqoritmin addımlarını təhlil edərkən, bu alqoritmin eksponensial alqoritm olduğunu aydın şəkildə görə bilərik. Həmçinin, K-medoids alqoritmi $NP-çətin$ sinfinə daxildir. Bu alqoritmin zaman mürəkkəbliyi aşağıdakı kimi təyin olunur:

$$O((n-k)^2 * m * k * t) \qquad (2.8)$$

Burada,

n – verilənlərin sayı

m – atributların sayı

k – yaradılan klasterlərin sayı

t – realizə olunan iterasiyaların sayı

kimi təyin olunur.

K-medoids alqoritminin müəyyən **üstünlükləri və çatışmazlıqları** mövcuddur.Bu alqoritmin **üstünlükləri** aşağıdakılardır:

1) K-medoids alqoritmi K-means alqoritminə nisbətən küy tipli və əhəmiyyətsiz verilənlərə qarşı daha dayanıqlıdır.
2) Klasterləşdirmə zamanı klasterlərin mərkəzlərini təmsil edən medoidlər, K-means alqoritmindən fərqli olaraq həqiqi verilənlərdir.

K-medoids alqoritminin **çatışmazlıqları** aşağıdakılardır:

1) K-means alqoritmində olduğu kimi, K-medoids alqoritminin də ən böyük çatışmazlığı onun başlanğıc klaster mərkəzlərinin seçilməsindən asılı olmasıdır.
2) Böyük həcmli verilənlərin analizi üçün K-medoids alqoritmindən istifadə olunması zaman mürəkkəbliyi səbəbindən səmərəli sayılmır.

K-means alqoritmi üçün qoyulmuş müəyyən məhdudiyyət şərtləri – dayanma, zaman və iterasiya sayı kriteriyaları eyni qayda ilə K-medoids alqoritmi üçün də qoyula bilər. Aydındır ki, qoyulmuş kriteriyalar klasterləşdirmənin keyfiyyətinə və məsələnin optimal həllinə birbaşa təsir edir.

K-medoids alqoritmini analitik təsviri zamanı, alqoritmin riyazi şərhinə baxacağıq. Ötən fəsildə, klasterləşdirmə probleminin optimallaşdırma məsələsinə gətirilməsi zamanı bir çox əsas anlayışlardan istifadə etmişdik. Beləliklə, K-medoids alqoritminin analitik təsvirinə baxaq.

Fərz edək ki, n sayda verilənlərdən ibarət olan sonlu, qabarıq və məhdud P=$\{X_1, X_2, X_3 \dots X_n\}$, $\forall X_i \in R^m$, $i = \overline{1,n}$ çoxluğu metrik (R^m, L_2) Hilbert fəzasında təyin olunmuşdur. Məqsədimiz metrik fəzada paylanmış verilənlərin k sayda elementdən ibarət olan

$$C = \{C_1, C_2, C_3 \dots C_k\}, \forall C_l \in R^m, l = \overline{1,k} \qquad (2.9)$$

klasterlər çoxluğunun elementlərində, yəni $C_1, C_2, C_3 \dots C_k$ klasterlərində qruplaşdırmaqdır. Məlumdur ki,

$$\forall\, C_l \subset P \text{ və } C_1 \cup C_2 \cup C_3 \cup \dots \cup C_k = P \qquad (2.10)$$

Həmçinin,

$$C_l \cap C_m = \emptyset\,,\ m = \overline{1,k} \text{ və } k \leq n \qquad (2.11)$$

zəruri şərti verilmişdir. Xüsusi $k = n$ halında hər bir element bir klasteri təmsil edir.

Klasterlərin medoidləri çoxluğunu aşağıdakı kimi göstərsək,

$$\mathrm{M} = \{m_1, m_2, m_3 \dots m_k\} \text{ və } \forall m_l \in C_l \subset P \qquad (2.12)$$

məlum olar.

K-medoids alqoritmi, baxılan P çoxluğundan elə k sayda medoid seçir ki, verilənlərdən medoidlərə qədər məsafələr cəmi minimum qiymət alsın [5]. Belə ki,

$$J = \sum_{i=1}^{n} \min_{m \in \mathrm{M}} \rho(m, X_i) \qquad (2.13)$$

məlum olur.

Ümumi halda, fərz etsək ki, optimal m^* medoidinin seçilməsi üçün ona məsafəcə yaxın X^* verilənləri məlumdur. Onda alqoritm, hər bir iterasiyanı aşağıdakı minimallaşdırma məsələsinin (2.14) üzərindən realizə edir [11]:

$$(m^*, X^*) = \underset{(m,X)\in \mathrm{M} \times (\mathrm{P}\setminus\mathrm{M})}{\arg\min} \frac{1}{n}\sum_{i=1}^{n} \min\left[\rho(X, X_i), \min_{m'\in \mathrm{M}\setminus\{\mathrm{m}\}} \rho(m', X_i)\right]$$

Aydındır ki, (m^*, X^*) cütü alqoritmin tapdığı optimal medoidi və onun ətrafında klasterləşdirilmiş verilənləri təmsil edir.

2.3. CLARA alqoritminin analizi

K-medoids alqoritminin ən geniş yayılmış modifikasiyalarından olan CLARA *(eng: Clustering for Large Application)* alqoritmi, ingilis abreviaturası ilə adlandırılmış və *"böyük ölçülü verilənlərin klasterləşdirməsi"* mənasına gəlir. CLARA alqoritmi ilk dəfə 1990-cı ildə *K-medoids* alqoritmini tədqiq etmiş *Leonard Kaufman* və *Peter J. Rousseeuw* tərəfindən təklif olunmuşdur.

CLARA alqoritminin əsasını klassik medoidlər ətrafında bölünməyə əsaslanan klasterizasiya alqoritmi təşkil edir. Lakin CLARA alqoritminin özünəməxsus xüsusiyyətləri mövcuddur. Belə ki, bu alqoritm verilənlər arasından müəyyən sayda altçoxluq seçir və nümunə olaraq seçilmiş hər bir alt çoxluq üzərində K-medoids alqoritmi realizə olunur. Seçilmiş medoidlər çoxluğundan minimum məsafəni doğuran *k* sayda medoid ətrafında klasterləşdirmə yekunlaşır.

CLARA alqoritminin mərhələlərini daha lokanik şərhi üçün bu alqoritmi mətn şəkildə təsvir edək [3]:

1) *n* sayda veriləni klasterləşdirmək üçün klasterlərin sayı – k təyin olunur.
2) Təsaüfi yolla *l* elementdən ibarət olan *s* sayda alt çoxluq seçilir.
3) Seçilmiş hər bir alt çoxluq üzərində K-medoids alqoritmi realizə olunur və hər bir alt çoxluq üçün *k* sayda medoid seçilir.

4) Hər bir verilən ona məsafəcə ən yaxın olan medoid ətrafında qruplaşdırılır.
5) Ən yaxın medoid ətrafında qruplaşdırılmış verilənlərlə medoid arasındakı məsafələr cəmi hesablanır.
6) Məsafələr cəminin minimum olduğu alt çoxluğun medoidlərini optimal medoidlər olaraq seçin.

CLARA alqortimi, digər tədqiq olunmuş alqoritmlər kimi $NP-çətin$ sinfinə daxildir. Bu alqoritmin zaman mürəkkəbliyi,

$$O(t*(p*k*(s-k)*m\ +p*k*(n-k)*m)) \qquad (2.15)$$

kimi tapılır. Burada,

n – verilənlərin sayı

m – atributların sayı

k – yaradılan klasterlərin sayı

t – realizə olunan iterasiyaların sayı

s – seçilmiş alt çoxluqların ölçüsü

p – seçilmiş alt çoxluqların sayı

kimi təyin olunur.

CLARA alqoritminin müəyyən **üstünlükləri və çatışmazlıqları** mövcuddur. Bu alqoritmin **üstünlükləri** aşağıdakılardır:

1) CLARA alqoritmi K-medoids və K-means alqoritminə nəzərən böyük həcmli verilənlərin analizi üçün səmərəli sayılır və klasterləşdirmə keyfiyyətli aparılır.
2) CLARA alqoritmi klasterləşdirmə prosesini K-medoids alqoritminə nəzərən daha qısa zaman ərzində optimal medoidləri seçərək sonlandırır.

CLARA alqoritminin **çatışmazlıqları** aşağıdakılardır:

1) CLARA alqoritmi ilə klasterləşdirmənin effektivliyi verilənlərin ölçüsündən asılıdır.

2) K-means və K-medoids alqoritmində olduğu kimi, CLARA alqoritminin də ən böyük çatışmazlığı onun başlanğıc klaster mərkəzlərinin seçilməsindən asılı olmasıdır.
3) Təsadüfi yolla seçilmiş alt çoxluqlar verilənlərin aşağı keyfiyyətli klasterləşdirilməsinə səbəb ola bilər.

Öncəki tədqiq olunmuş alqoritmlər üçün qoyulmuş müəyyən məhdudiyyət şərtləri – dayanma, zaman və iterasiya sayı kriteriyaları eyni qayda ilə CLARA alqoritmi üçün də qoyula bilər. Həmçinin təsadüfi yolla seçiləcək alt çoxluqların sayı da bu kriteriyalar arasına daxil edilə bilər. Aydındır ki, qoyulmuş kriteriyalar klasterləşdirmənin keyfiyyətinə və məsələnin optimal həllinə birbaşa təsir edir.

2.4. CLARANS alqoritminin analizi

CLARANS *(eng: Clustering Large Applications based on Randomized Search)* alqoritmi, ingilis abreviaturası ilə adlandırılmış və *"təsadüfi axtarışa əsaslanaraq böyük ölçülü verilənlərin klasterləşdirməsi"* mənasına gəlir [9]. CLARANS alqoritmi K-medoid alqoritminin modifikasiyası olub, klasterləşdirmə zamanı dəqiqliyi və keyfiyyəti ilə digər modifikasiyalardan fərqlənir. K-medoids və CLARA alqoritmlərinin konseptual olaraq bir çox xüsusiyyətlərini özündə daşıyan CLARANS alqoritminin CLARA alqoritmindən əsas fərqi və məxsusi özəlliyi odur ki, CLARA alqoritmində öncədən seçilmiş müəyyən alt çoxluqlar üçün iterasiyalar realizə olunurdusa, bu alqoritm hər iterasiya zamanı təsadüfi yolla seçilmiş yeni alt çoxluq üçün prosesi realizə edir [6]. Bu isə, hər növbəti iterasiyada alt çoxluqların optimal seçimi üçün uyğun şərait yaradır.

Qeyd edək ki, hər növbəti iterasiyada seçilən yeni alt çoxluğun elementləri öncəki iterasiyada seçilmiş medoidlərə qonşu olan verilənlər olur. Buna görə də alqoritmi realizə etmədən öncə seçiləcək maksimal qonşu olan verilənlərin sayını qeyd etməliyik.

CLARANS alqoritminin mərhələlərini daha lokanik şərhi üçün bu alqoritmi mətn şəkildə təsvir edək [14]:

1) n sayda veriləni klasterləşdirmək üçün klasterlərin sayını – k `nı təyin edin.
2) Təsadüfi yolla k sayda medoid seçin.
3) Seçilmiş medoidlərə qonşu olan verilənlərin sayını – *maks_qonshu* təyin edin.
4) Əgər m medoidinə qonşu olan hər hansısa bir X verilənindən yerdə qalan bütün verilənlərə qədər olan məsafələr cəmi, baxılan m medoidinin doğurduğu məsafələr cəmindən kiçikdirsə, X veriləni həmin m medoidi ilə əvəz olunur.
5) Hər bir iterasiya üçün (4) addımı realizə olunur və optimal medoidlər seçilir.

CLARANS alqortimi, digər tədqiq olunmuş alqoritmlər kimi NP-çətin sinifinə daxildir.Bu alqoritmin zaman mürəkkəbliyi,

$$O(t * n * m * k * maks\_qonshu) \qquad (2.16)$$

kimi tapılır.Burada,

n – verilənlərin sayı

m – atributların sayı

k – yaradılan klasterlərin sayı

t – realizə olunan iterasiyaların sayı

maks_qonshu – alt çoxluğun təyini zamanı təsadüfi yolla medoidlər ətrafında seçiləcək maksimal qonşu verilənlərin sayı

kimi təyin olunur.

CLARANS alqoritminin müəyyən **üstünlükləri və çatışmazlıqları** mövcuddur. Bu alqoritmin **üstünlükləri** aşağıdakılardır [14]:

1) CLARANS alqoritmi K-means, K-medoids və onun digər modifikasiyası olan CLARA alqoritmi ilə müqayisədə ən keyfiyyətli klasterləşdirilməni realizə edir.

2) CLARANS alqoritmi ilə klasterləşdirmə zamanı əhəmiyyətsiz verilənlər klasterlərə aid edilmir.

CLARANS alqoritminin **çatışmazlıqları** aşağıdakılardır:

1) CLARANS alqoritmi ilə klasterləşdirmənin keyfiyyəti baxılan verilənlərin bazasının həcmindən bilavasitə asılıdır.
2) Medoidlərə qonşu olan verilənlərin qeyri-dəqiq seçilməsi verilənlərin aşağı keyfiyyətlə klasterləşdirilməsinə səbəb ola bilər.

Tədqiq olunmuş alqoritmlər üçün qoyulmuş müəyyən məhdudiyyət şərtləri – dayanma, zaman və iterasiya sayı kriteriyaları eyni qayda ilə CLARANS alqoritmi üçün də qoyula bilər. Həmçinin hər iterasiyada alt çoxluğun təyini zamanı təsadüfi yolla seçiləcək maksimal qonşu verilənlərin sayı da bu kriteriyalar arasına daxil edilə bilər. Aydındır ki, qoyulmuş kriteriyalar klasterləşdirmənin keyfiyyətinə və məsələnin optimal həllinə birbaşa təsir edir.

III FƏSIL . K-MEANS, K-MEDOİDS, CLARA VƏ CLARANS ALQORİTMLƏRİNİN PYTHON MÜHİTİNDƏ EKSPERİMENTAL MÜQAYİSƏSİ

3.1. Python mühitinin qurulması və əsas komponentlər üsulu

Müasir proqramlaşdırma dillərinin hər biri istifadə olunma məqsədi, səmərəliliyi ilə bir-birindən fərqlənərək müxtəlif sahələrə tətbiq olunur. Verilənlər elmində, bu elmin davamçısı olan süni intellekt və maşın öyrənmə mühəndisliyinin də geniş istifadə etdiyi dil **Python** proqramlaşdırma dilidir. *Python* proqramlaşdırma dili riyazi yönümlü və rahat sintaksisli olması ilə verilənlər elminin məşğul olduğu sahələrə tətbiqi, proqramçılar arasında geniş yayılmışdır.

Həmçinin hal-hazırda bir neçə platforma proqram və maşın öyrənməsi mühəndisləri üçün açıq qaynaqlı verilənlər bazasını *(eng: open-source databases)* təqdim edir. Qlobal dünyada maşın öyrənməsi mühəndisləri arasında geniş yayılmış belə platformalardan biri də *1987*-ci ildə *Kaliforniya Universitetinin* aspirantı *David Aha* və onun gənc həmkarları tərəfindən yaradılan *UCI –Machine Learning Repository* platformasıdır. Bu platforma özündə yüzlərlə verilənlər anbarlarının, verilənlər bazalarını ehtiva edir.

Tədqiq olunan alqoritmlərlə aparılmış eksperimentlər baxılan məsələnin verilənlərini özündə ehtiva edən *.csv (eng-abr: comma-separated values – vergüllə ayrılmış qiymətlər adlı faylı)* uzantılı, *text/csv* formatlı fayldan istifadə edəcəyik. Fayl sətir və sütunlardan ibarət olub, cədvəlin hər xanası atributların qiymətlərini özündə saxlayır. Eksperimentlər *UCI* platformasından endirdiyimiz ***station.csv*** faylının verilənləri üzərində Pyhton mühitində aparılmışdır. Baxılan *station.csv* faylındakı verilənlər binaların adlarını, nömrələrini, son demoqrafik məlumatlara əsasən həmin binada yaşayan əhalinin sayını, əhalinin sıxlığını, aktiv şəkildə telekommunikasiya texnologiyalarından istifadə edən əhalinin orta yaşı və son il üçün realizə olunmuş telekommunikativ tranzaksiyaların tezliyi adı altında atributlaşdırılmışdır. Məqsədimiz

məhdud sayda yerləşdiriləcək baza stansiyalarının klasterləşdirmə alqoritmləri ilə uyğun binaların seçilməsini təmin etməkdir.

Tədqiq edilmiş alqoritmləri qiymətləndirmək üçün Intel Core – i5, 2.50 Ghz, 8 Gb RAM xarakteristikalarına malik kompüterdə Python proqramlaşdırma dilindən istifadə edilmişdir. İşçi mühiti formalaşdırmaq üçün *inteqrasiya edilmiş mühit - IDE (eng: Integrated Development Environment)* seçilməlidir. Bunun üçün ***JupyterLab*** adlı onlayn kompilyator rolunu oynayan işçi mühitini seçəcəyik. İşçi mühitdə nəticələrin ardıcıl kompilyasiyasını, yəni yazılmış kod fraqmentinin nəticəsini görmək üçün *JupyterLab* işçi mühitində açılmış yeni pəncərədə *qeyd dəftərçəsi (eng: JupyterLab Notebook)* açırıq. Açılmış pəncərədə yazılacaq kodlar *ipykernel Python 3* `də - onlayn kompilyasiya olunur.

Alqoritmlərlə eksperimentlər aparılan zaman Python proqramlaşdırma dilinin bir neçə kitabxanaları bu mühitə əlavə edilmişdir. Əlavə olunacaq *(eng: import)* kitabxanalar verilənlərinin vizualizasiyası və verilənlər üzərində rahat işləmək imkanı yaradır.

JupyterLab işçi mühitinə əlavə olunacaq Python proqramlaşdırma dilinin **kitabxanalar**ına baxaq:

1) *Pandas* – verilənlərin ilkin emalı və analizi üçün istifadə olunan bu kitabxana, verilənlər üzərində müəyyən əməliyyatları aparmaq üçün özündə bir neçə metodları, funksiyaları ehtiva edir [12]. Pandas kitabxanası *.csv*, *JSON* uzantılı fayllar və *SQL* verilənlər bazası cədvəlləri ilə işləmək imkanı yaradır, həmçinin bu uzantıya malik faylları dəstəkləyir.
2) *NumPy* – böyük həcmli verilənlərin analizi zamanı onların üzərində bir neçə riyazi əməliyyatların icra edilməsi üçün istifadə olunan kitabxanadır. Numpy kitabxanasında verilənlər massiv formasında təsvir olunur.

3) *Matplotlib* – verilənlərin müstəvidə və üç ölçülü fəzada vizuallaşdırılması, histoqramların, xətti asılılığın qurulması və s. üçün istifadə olunur.
4) *Sci-kit Learn* – müasir maşın öyrənməsində ən geniş istifadə olunan kitabxanalardandır. Bu kitabxanda, klassifikasiya, reqressiya və klasterləşdirmə üsullarına məxsus funksiyalardan istifadə oluna bilər.

Məlumdur ki, böyük ölçülü verilənlərin vizuallaşdırılması zamanı həmin verilənlərin müstəvi üzərində yerləşdirmək mümkün olmur. Python mühitində ekperimentlərin aparılması zamanı bu problemi aradan qaldırmaq üçün *Sci-kit Learn* kitabxanasının *PCA* adlı modulundan istifadə edəcəyik. Bu modul, böyük ölçülü verilənlərin iki və ya üç ölçülü fəza üzərində vizuallaşdırılması üçün istifadə olunur.

PCA *(eng: Principial Component Analysis)* - əsas komponentlər üsulu olaraq adlandırılır və qeyd etdiyimiz kimi, böyük ölçülü fəzada yerləşdirilmiş verilənlərin daha aşağı ölçülü fəzaya köçürülməsi üsuludur. İlk dəfə *1901*-ci ildə *Karl Peaarson* tərəfindən tətbiq edilmiş bu üsul bəzən *Hotelling çevirməsi* olaraq da adlandırılır [19]. Ekonometrika, riyazi statistika, verilən elmi və maşın öyrənməsi kimi istiqamətlərdə tətbiq olunur.

Məlumdur ki, verilənlər cədvəl *(eng: tabular data)* formasında təqdim olunarkən, bu cədvəlin sütunlarında yerləşdirilmiş əlamət vektoru əksər hallarda daha böyük ölçülü vektor olur. Bu halda isə, istər ilkin emal zamanı, istərsə də istənilən alqoritmin tədqiqi zamanı həmin verilənləri müstəvidə vizuallaşdırmaq mümkün olmur. Əsas komponentlər üsulu, məhz bu probleminin həllini özündə ehtiva edir.

Bu üsulda, verilənlərin məlum bazis üzrə səpələnməsinə baxılır. Səpələnmənin daha kiçik olduğu, dispersiyanın daha böyük olduğu istiqamətdə cari verilənlər üçün yeni bazis müəyyən olunur. Prosesi daha aydın başa düşmək üçün riyazi şərhinə baxaq.

Fərz edək ki, n sayda veriləndən ibarət olan sonlu $P=\{X_1, X_2, X_3 \dots X_n\}$, $\forall X_i \in R^m$, $i = \overline{1,n}$ çoxluğu *m* ölçülü *Hilbert* fəzasında təyin olunmuşdur. Həmçinin,

$$P = \left(X_1, X_2, X_3 \dots X_m\right) = \begin{Vmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1m} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2m} \\ x_{31} & x_{32} & x_{23} & \cdots & x_{3n} \\ \vdots & & \cdots & & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nm} \end{Vmatrix} = \left\|x_{ij}\right\|_{n\,x\,m} \quad (3.1)$$

Burada i - verilənin nömrəsini, j – verilənin fəza ölçüsünü təmsil edir.

Məlum P koordinatlar matrisini müəyyən R^m fəzasında təyin olunmuş $n\,x\,m$ ölçülü Y matrisinə xətti çevirmək üçün, elə $n\,x\,n$ ölçülü A xətti çevirməsinin matrisi var ki,

$$Y = A(P) \quad (3.2)$$

kimi ifadə olunsun.

Xətti çevrilməni realizə etmək üçün məlum P koordinatlar matrisini və A xətti çevirməsinin simmetrik matrisini təmsil edən sütun və sətir vektorlarını təyin edək.Bunlar,

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{pmatrix} \text{ və } P = \left(X_1, X_2, X_3 \dots X_m\right) \quad (3.3)$$

kimi təyin olunmuşdur. Onda, (3.4) xətti çevrilməsinə baxsaq,

$$Y = A(P) = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{pmatrix} x \left(X_1, X_2, X_3 \dots X_m\right) = \begin{Vmatrix} A_1.X_1 & A_1.X_2 & \cdots & A_1.X_m \\ A_2.X_1 & A_2.X_2 & \cdots & A_2.X_m \\ A_3.X_1 & A_3.X_2 & \cdots & A_3.X_m \\ \vdots & \cdots & \cdots & \vdots \\ A_n.X_1 & A_n.X_2 & \cdots & A_n.X_m \end{Vmatrix}$$

Qeyd edək ki, burada $A_i.X_j$ – işarəsi həmin vektorların skalyar hasilini təmsil edir.

Xətti çevrilmiş P matrisinin yeni bazisləri və komponentləri A matrisinin sətirlərini təmsil edir. Lakin, aydındır ki, A matrisi və onun ən uyğun sətirlərinin, komponentin, başqa sözlə desək, obyektin ilkin bazis vektorları ilə yeni bazis vektorları arasındakı ən uyğun asılılığı xarakterizə etmək üçün kovariasiya anlayışından istifadə

edəcəyik. Aydındır ki, kovariasiya anlayışı birbaşa olaraq dispersiya anlayışı ilə əlaqəlidir.

Qurulmuş kovariyasiya matrsinin məxsusi ədədləri tapılır və ən böyük qiymətə malik məxsusi ədəd seçilir. Seçilmiş məxsusi ədədə uyğun olaraq, məxsusi vektor tapılır. Vektor normallaşdırıldıqdan sonra, yeni bazis olaraq təyin olunur.Təyin olunmuş bazis üzrə obyektlər yerləşdirilir.

Ümumi halda, aşağıda qurulmuş (3.5) kovariyasiya matrisi üzərində müəyyən çevirmələr etsək [4],

$$cov_Y = \frac{1}{m-1} YY^T = \frac{1}{m-1}(AP)(AP)^T = \frac{1}{m-1}(AP)(A^T P^T) = \frac{1}{m-1} A(PP^T)A^T$$

alarıq.

Məlumdur ki, $\forall S$ simmetrik matrisi üçün $S = S^T$. Buna görə də, $PP^T = (P^T)^T(P)^T = (PP^T)^T$, olduğu üçün $S_{n\,x\,n} = PP^T$ əvəzləməsi apara bilərik. Onda,

$$cov_Y = \frac{1}{m-1} ASA^T \qquad (3.6)$$

nəticəsini alarıq.

Həmçinin aydındır ki, hər kvadrat simmetrik matris, ortoqonal matris, ortoqonal matrisin transponirə olunmuş matrisi və dioqanal matrisin hasilinə bərabərdir. Onda, $\exists Q_{n\,x\,n} = Q^T Q^{-1}$ ortoqonal matrisi və $\exists D$ dioqanal matrisi var ki,

$$S = QDQ^T \qquad (3.7)$$

kimi ifadə olunsun. Burada,

1) Q ortoqonal matrisinin sütun vektorları S matrisinin məxsusi vektorlarıdır;
2) D dioqanal matrisi S matrisinin məxsusi ədədlərindən təşkil olunmuşdur.
3) S matrisinin xətti asılı olmayan sütun vektorlarının sayı, yəni ranqı, S-in ortoqonal vektorlarının sayıdır.

Xətti çevrilmə üçün uyğun matrisin seçilməsi zamanı, xüsusi halda *S* matrisinin məxsusi vektorlarını seçəcəyik. Belə ki, $A = Q^T$ və aydındır ki,

$$cov_Y = \frac{1}{m-1} ASA^T = \frac{1}{m-1} Q^T (QDQ^T)Q \qquad (3.8)$$

Məlumdur ki, Q ortoqonal matris olduğu üçün, $QQ^T = I$ ifadəsi doğrudur. Burada I – vahid matrisdir. Onda,

$$cov_Y = \frac{1}{m-1} D = \frac{1}{m-1} \begin{bmatrix} \lambda_{11} & \cdots & \lambda_{1m} \\ \vdots & \ddots & \vdots \\ \lambda_{m1} & \cdots & \lambda_{mm} \end{bmatrix} \qquad (3.9)$$

Hər bir λ_{jj} məxsusi ədədlərinə uyğun e_j vektoru məxsusi vektor sayılıb, yeni təyin olunmuş bazis vektorlarıdır və obyektlər yeganə qayda ilə yeni bazis üzrə ayırmaq mümkündür.

Ümumi halda, məlumdur ki, bu məxsusi vektorların tapılması zamanı

$$cov_Y E = \lambda E, E \neq 0 \text{ və } \lambda_{jj} \in E \qquad (3.10)$$

bərabərliyindən alınan bircins xətti tənliklər sisteminin sıfırdan fərqli həllinin varlığı üçün zəruri və kafi şərt onun məchullarının əmsallarından qurulmuş matrisinin

$$det|cov_Y - \lambda I| = 0 \qquad (3.11)$$

şərtini ödəməsidir. Tapılmış hər bir λ_{jj} məxsusi ədədi üçün məxsusi vektor isə yuxarıdakı bərabərlikdən tapılır.

Yeni bazisə köçürülmə zamanı ilkin bazisin tapılması zamanı böyük məxsusi ədədin məxsusi vektoru tapılır. Daha sonra köçürülmüş fəzanın ölçüsünə əsasən digər bazislərin tapılması üçün növbəti kiçik ədədi qiymətə malik məxsusi ədədin məxsusi vektoruna baxılır.

3.2. K-means alqoritminin Python mühitində aparılmış eksperimenti

İlk növbədə *JupyterLab* işçi mühitinə *Pandas, NumPy, Matplotlib, Sci – kit Learn* kitabxanalarını, alqoritmin realizə olunma müddətini ölçmək üçün *timer* modulunu qoşuruq və *start = timer()* dəyişəni ilə zamanın ölçülməsinin başlanmasını qeyd edirik. Faylın mühitə əlavə olunması üçün *Pandas* kitabxanasının *.read_csv()* metodunu işlədirik:

```
>> import pandas as pd
>> import numpy as np
>> import matplotlib.pyplot as plt
>> from sklearn.decomposition import PCA
>> from timeit import default_timer as timer

start = timer()

# Verilənlərin mühitə əlavə olunması
data = pd.read_csv('station.csv')
```

Əlavə olunmuş *station.csv* faylındakı verilənlər, *data* adlı dəyişənə cədvəl formasında mənimsədilir. Nəticəni və ilkin emal olunmamış cədvəli görmək üçün *data* dəyişənini yazıb *"Run"* əmrini verə bilərik *(Şək.3.1.):*

[68]:

	Bina	Binanin_nomresi	Ehalinin_sayi	Ehalinin_sixligi	Aktiv_orta_yas	Tranzaksiya_tezliyi
0	Lukken	284	616	11	59	3
1	Roxbury	2	9351	56	51	80
2	Dakota	742	18129	51	50	20
3	Pennsylvania	9	1086	68	64	37
4	Dapin	937	6555	3	32	35
...	...	...	...	...	...	...
995	Kinsman	1917	9520	13	35	52
996	Coleman	13902	15311	13	28	28
997	Meadow Valley	78	14244	18	18	42
998	Rusk	227	11191	85	29	55
999	Paget	1669	18810	50	34	94

1000 rows × 6 columns

Şək 3.1. *station.csv* faylındakı verilənlərin cədvəl formasında təsviri

Verilənlərin ilkin emalı zamanı cədvəli lokanik hala gətirmək üçün, başqa sözlə desək, seçəcəyimiz atributlara əsasən yeni cədvəl formasına salmaq üçün atributların

adlarından ibarət bir əlamət massivi qurmalıyıq. Bu massivin elementlərinin sayı verilənlərin yerləşdiyi fəzanın ölçülərini təmsil edir. Hər bir verilən *atr* massivinin elementlərinin sayında – 4 ölçülü fəzada təyin olunmuşdur.

Cədvəldən yalnızca *atr* massivində təyin olunmuş elementlərlə eyni adda olan olan atributların yerləşdiyi sütunları *data[atr]* kodu ilə yeni cədvələ daxil etməsini istəyirik. Daha sonra yalnızca seçilmiş atributları özündə saxlayan yeni bir alt cədvəli *data* adlı dəyişənin daxilində *copy*() metodu vasitəsi ilə saxlayacağıq. *data* dəyişəni üzərində riyazi əməliyyatlar apara bilməyimiz üçün onu massiv tipinə çevirərkən *NumPy* kitabxanasının *.to_numpy()* metodundan istifadə edəcəyik:

```
atr=["Ehalinin_sayi","Ehalinin_sixligi","Aktiv_orta_yas","Tranzaksiya_tezliyi"]
data = data[atr].copy()
data = data.to_numpy()
```

Alqoritmi realizə etmədən öncə klasterləşdirmə zamanı klasterlərin sayını və realizə olunacaq maksimal iterasiya sayını daxil etməliyik.Bunun üçün,

```
# Klasterlerin sayı

k = 3

# Realizə olunacaq maksimal iterasiyaların sayı

maks_iterasiya = 100
```

Daha sonra, təsadüfi yolla mərkəzlərin təyin olunması zamanı *NumPy* kitabxanasının *np.random.choice()* metodundan istifadə edəcəyik ki, bu metodun daxilindəki *data.shape[0], k, replace=False* dəyişənləri cədvəlin sətir massivləri arasından təkrar olunmamaq şərti ilə daxil edilmiş klasterlərin sayı qədər verilənlər seçir. Bu verilənlərin hər biri *list* tipli veriləndir:

```
merkezler = data[np.random.choice(range(len(data)), k, replace=False)]
```

İterasiyalar üçün *for i in range(maks_iterasiya):* dövrü qurulur və hər bir iterasiyada mərkəzlərlə verilənlər arasındakı məsafələr *np.linalg.norm()* metodu vasitəsi ilə hesablanır və minimum məsafəni doğuran verilənlər *np.argmin()* metodu vasitəsi ilə eyni nömrəli klasterlərə aid edilir. Hər növbəti iterasiyada klasterlərin yeni mərkəzləri seçilərkən *.mean()* metodu vasitəsi ilə verilənlər üçün statistik ədədi orta tapılır. Klasterlər üçün seçilmiş yeni mərkəzlər dəyişmirsə, o zaman, alqoritm sonlandırılır:

```
for i in range(maks_iterasiya):
    # Hər bir veriləni ona yaxın mərkəz ətrafında qruplaşdır
    mesafeler= np.linalg.norm(data[:, np.newaxis] - merkezler, axis=2)
    klaster_sinifleri = np.argmin(mesafeler, axis=1)
    # Mərkəzləri yenilə
    yeni_merkezler = np.array([data[klaster_sinifleri == j].mean(axis=0) for j in range(k)])
   if np.all(merkezler == yeni_merkezler):
       break
    merkezler = yeni_merkezler
```

Klasterləşdirmənin keyfiyyətini ölçmək üçün *SSE (eng: sum of squared errors)* anlayışından istifadə edəcəyik və hər bir iterasiya üçün *SSE* qiymətini ekrana çıxaracağıq:

```
# SSE qiymətini hesabla

    sse = np.sum((data - merkezler[klaster_sinifleri]) ** 2)

    print(f'İterasiya {i+1}, SSE: {sse}')
```

Yuxarıda qeyd etdiyimiz kimi, böyük ölçülü verilənlərin vizuallaşdırılması üçün Sci-kit Learn kitabxanasının PCA modulundan istifadə etməliyik. Verilənləri müstəvidə vizuallaşdırmak üçün təyin olunmuş *PCA(n_components=2)* metodundan istifadə edirik və *pca.fit_transform(data)* metodu ilə verilənləri miqyaslandırırıq.

```
# PCA ilə vizuallaşdır
pca = PCA(n_components=2)
data_pca= pca.fit_transform(data)
merkezler_pca = pca.transform(merkezler)
```

Verilənlərin vizuallaşdırılması zamanı *Matplotlib* kitabxanasının *.scatter()* metodundan istifadə edəcəyik. Verilənləri və klaster mərkəzlərini vizual olaraq fərqləndirmək üçün *marker='*', s=200, linewidths=3, color='r', zorder=10* qiymətlərindən istifadə edəcəyik. *plt.title('KMeans alqoritmi')* metodu ilə qrafikimizin başlığını təyin edirik və *plt.show()* metodu ilə vizuallaşdırmanı gerçəkləşdiririk.

```
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=klaster_sinifleri)
plt.scatter(merkezler_pca[:, 0], merkezler_pca[:, 1], marker='*', s=200,
linewidths=3, color='r', zorder=10)
plt.title('K-means alqoritmi')
plt.show()
```

Vizuallaşdırma mərhələsindən sonar klasterlərin mərkəzlərini və alqoritmin realizə olunması müddətini görmək üçün *print('Son mərkəzlər:', merkezler)* və *timer()* metodlarından istifadə edirik:

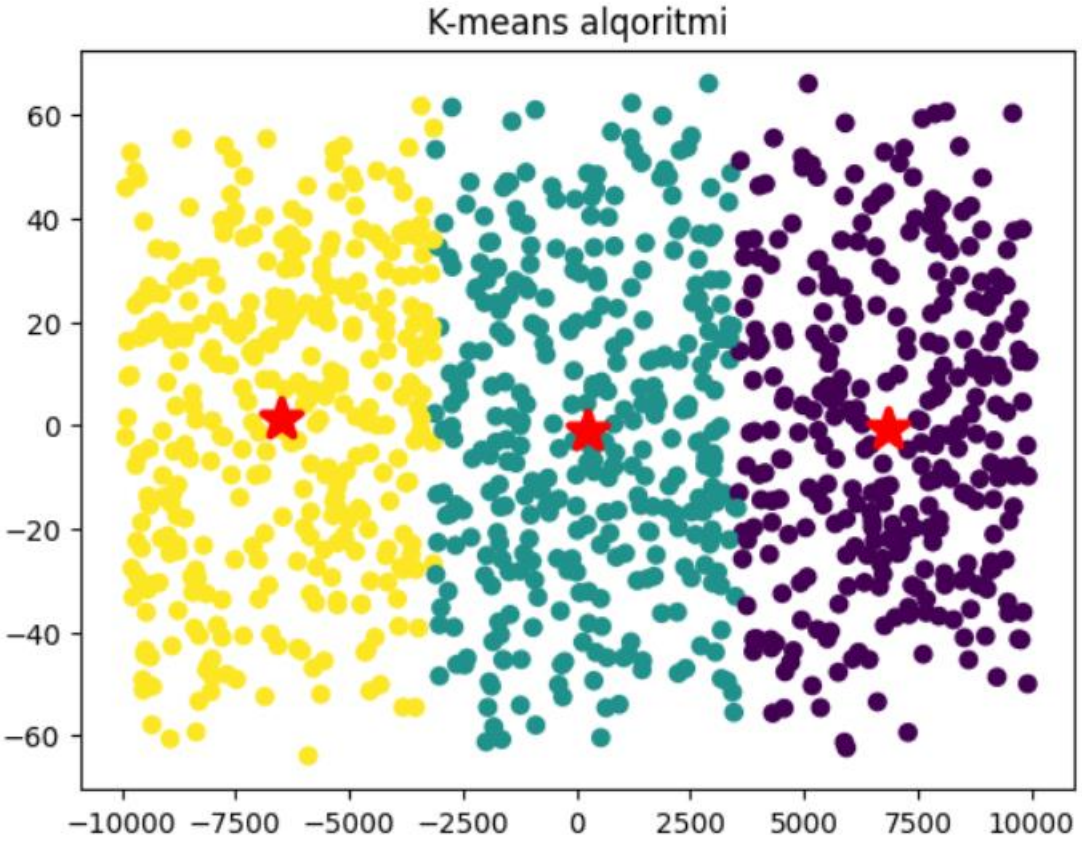
```
# Son mərkəzləri ekrana çıxart
print('Son mərkəzlər:', merkezler)
end = timer()
print(f"Realizə olunma müddəti: {end - start} saniyə")
```

Beləliklə, aparılmış eksperimentin nəticəsini analiz etmək üçün *Run* əmri ilə alqoritmi realizə etsək, iterasiyaların nömrəsini və hər bir iterasiya zamanı klasterləşdirmənin keyfiyyətini *Cədvəl 3.1*`də görə bilərik:

İterasiya 1, SSE: 6968214021.505647	*İterasiya 11, SSE: 3739683023.6146812*
İterasiya 2, SSE: 4802860469.731585	*İterasiya 12, SSE: 3738594628.571469*
İterasiya 3, SSE: 4280397949.683489	*İterasiya 13, SSE: 3737677283.036748*
İterasiya 4, SSE: 3979444354.6069856	*İterasiya 14, SSE: 3737244886.5957556*
İterasiya 5, SSE: 3837864549.2173023	*İterasiya 15, SSE: 3736815818.935952*
İterasiya 6, SSE: 3789038191.3996024	*İterasiya 16, SSE: 3736413170.929201*
İterasiya 7, SSE: 3773740690.6632257	*İterasiya 17, SSE: 3736326785.722805*
İterasiya 8, SSE: 3760942477.03261	*İterasiya 18, SSE: 3736201983.1367464*
İterasiya 9, SSE: 3745630923.9700923	*İterasiya 19, SSE: 3736132354.2168484*
İterasiya 10, SSE: 3742174122.5810328	

Cədvəl 3.1. K-means alqoritmi ilə klasterləşdirmə zamanı realizə olunmuş iterasiyalar

K-means alqoritmi ilə klasterləşdirmə zamanı verilənlərin, klasterlərin mərkəzlərinin, o cümlədən, cari klasterlərə aid edilmiş verilənlərin vizullaşdırılması *Şək. 3.2*`də göstərilmişdir. Klasterlərdə yerləşdirilmiş elementlər uyğun olaraq, *sarı*, *yaşıl* və *bənövşəyi* rənglərdə, klasterlərin mərkəzləri isə *qırmızı ulduz* işarələməsi ilə vizuallaşdırılmışdır:



Şək. 3.2. K-means alqoritmi ilə klasterləşdirmənin vizuallaşdırılması

Klasterləşdirmə $T^* = 0.17726159999983793$ saniyə *(Şək. 3.3)* müddətində həyata keçirilmişdir. Proses $I_{opt} = 19$ nömrəli iterasiyada sonlandırılmış və optimal iterasiyaya uyğun $SSE = 3736132354.2168484$ dəyəri məlum olmuşdur.

```
Son mərkəzlər: [[ 3225.67192429    48.67192429    38.10094637    50.48895899]
 [ 9795.74183976    49.16023739    40.74777448    49.32047478]
 [16548.44508671    52.63294798    38.39017341    48.06647399]]
Realizə olunma müddəti: 0.17726159999983793 saniyə
```

Şək. 3.3. Klasterlərin mərkəzləri və alqoritmin realizə olunma müddəti

Baza stansiyalarının yerləşdirilməsi məsələsində, K-means alqoritmi ilə klasterləşdirmə zamanı klaster mərkəzlərini təmsil edən verilənlər heç bir həqiqi əlamətlərə malik binanı təmsil etmir *(Şək. 3.4.)*. Qeyd edək ki, K-means alqoritmi ilə aparılan eksperiment zamanı Python mühitində yazılmış kod fraqmentləri *Əlavələr (1)* bölməsinə əlavə olunmuşdur.

```
merkezler_cedvel = pd.DataFrame(merkezler, columns=atr)
print("Son merkezler:\n", merkezler_cedvel)
```

```
Son merkezler:
    Ehalinin_sayi  Ehalinin_sixligi  Aktiv_orta_yas  Tranzaksiya_tezliyi
0    3225.671924         48.671924       38.100946            50.488959
1    9795.741840         49.160237       40.747774            49.320475
2   16548.445087         52.632948       38.390173            48.066474
```

Şək. 3.4. Klaster mərkəzlərinin əlamətləri

3.3. K-medoids alqoritminin Python mühitində aparılmış eksperimenti

Ötən yarım fəsildə K-means alqoritmi ilə klasterləşdirmə zamanı klaster mərkəzlərini təmsil edən verilənlər heç bir həqiqi əlamətlərə malik binanı təmsil etmədiyi üçün optimal həllin K-medoids alqoritmi ilə araşdıraq. K-means alqoritmi ilə apardığımız eksperimentdə olduğu kimi işçi mühitə uyğun kitabxanaların qoşulması, verilənlərin ilkin emalı, həmçinin klasterləşdirmə və vizuallaşdırılma mərhələləri K-medoids alqoritmi üçün eyni qayda ilə yerinə yetirilir.

Kitabxanaların qoşulması və verilənlərin ilkin emalı mərhələlərindən sonra K-medoids alqoritmi üçün klasterlərin sayını – *k=3* və realizə olunacaq maksimal iterasiya sayını - *maks_iterasiya = 100* təyin edirik.

Təsadüfi yolla medoidlərin təyini olunması zamanı *NumPy* kitabxanasının *np.random.choice()* metodundan istifadə edəcəyik ki, bu metodun daxilindəki *data.shape[0], k, replace=False* dəyişənləri cədvəlin sətir massivləri arasından təkrar olunmamaq şərti ilə daxil edilmiş klasterlərin sayı qədər verilənlər seçir [2]. Bu verilənlərin hər biri list tipli veriləndir:

```
# Təsadüfi yolla medoidlərin seçilməsi

medoidler = data[np.random.choice(data.shape[0], k, replace=False)]
```

İterasiyalar üçün *for i in range(maks_iterasiya):* dövrü qurulur və hər bir iterasiyada medoidlərlə verilənlər arasındakı məsafələr *np.linalg.norm()* metodu vasitəsi ilə hesablanır və minimum məsafəni doğuran verilənlər *np.argmin()* metodu vasitəsi ilə eyni nömrəli klasterlərə aid edilir. Hər növbəti iterasiyada seçilmiş medoid, *for j in range(k)* dövrü boyunca hər bir verilənlərlə əvəz olunur. Əvəzetmə zamanı klasterlərin keyfiyyəti ölçülür və ən keyfiyyətli klasterləşdirməni doğuran verilənlər yeni medoidlər olaraq təyin olunur. Klasterlər üçün seçilmiş yeni mərkəzlər dəyişmirsə, o zaman, alqoritm sonlandırılır:

```
for i in range(maks_iterasiya):
    # Hər bir veriləni ona yaxın mərkəz ətrafında qruplaşdır
    mesafeler = np.linalg.norm(data[:, np.newaxis] - medoidler, axis=2)
    klaster_sinifleri = np.argmin(mesafeler, axis=1)
    # Medoidləri yenilə
    yeni_medoidler = np.array([data[klaster_sinifleri == j][np.argmin(mesafeler[klaster_sinifleri == j].sum(axis=1))] for j in range(k)])
    if np.all(medoidler == yeni_medoidler):
        break
    medoidler = yeni_medoidler
```

Klasterləşdirmənin keyfiyyətini ölçmək üçün *SSE (eng: sum of squared errors)* anlayışından istifadə edəcəyik və hər bir iterasiya üçün *SSE* qiymətini ekrana çıxaracağıq:

```
# SSE qiymətini hesabla
    sse = np.sum((data - merkezler[klaster_sinifleri]) ** 2)
    print(f'İterasiya {i+1}, SSE: {sse}')
```

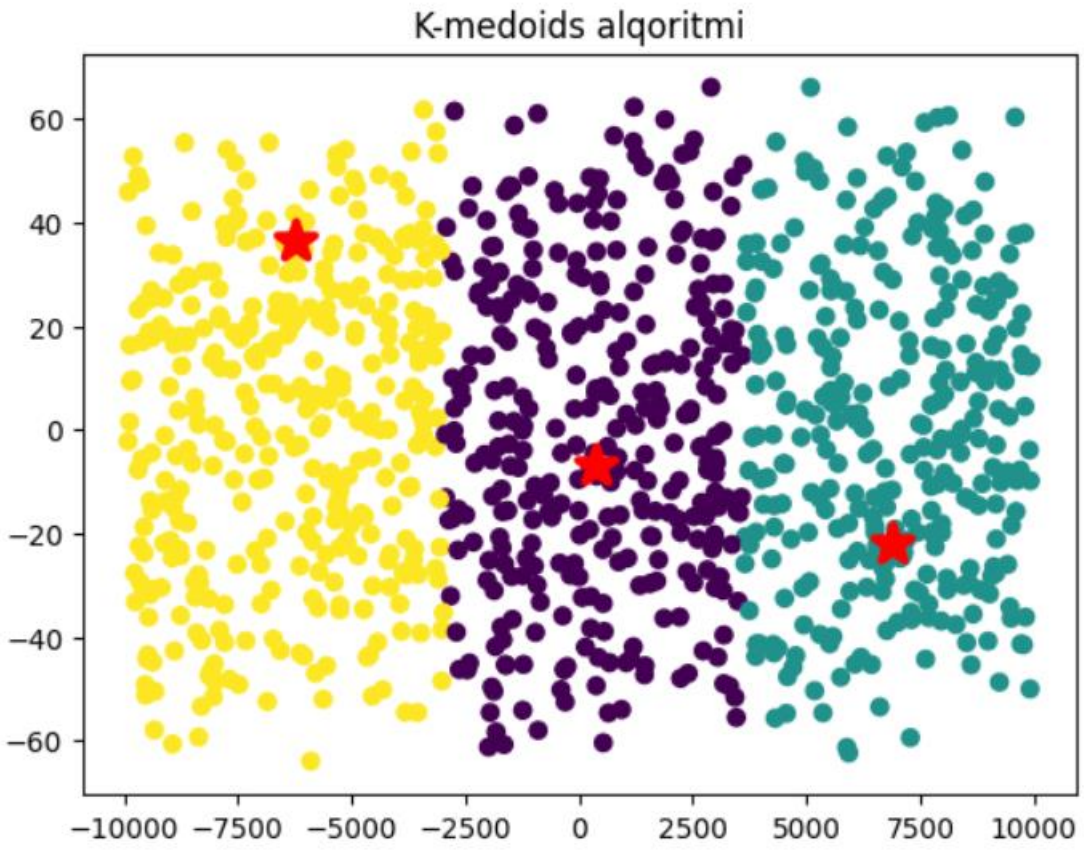
Vizuallaşdırılma mərhələsi K-means alqoritmi ilə apardığımız eksperimentdə olduğu kimi K-medoids alqoritmi üçün də eyni qayda ilə yerinə yetirilir.

Eksperimentin nəticəsini analiz etmək üçün *Run* əmri ilə alqoritmi realizə etsək, iterasiyaların nömrəsini və hər bir iterasiya zamanı klasterləşdirmənin keyfiyyətini *Cədvəl 3.2.*`də görə bilərik:

```
İterasiya 1, SSE: 10965131173
İterasiya 2, SSE: 20295881792
İterasiya 3, SSE: 29981762331
İterasiya 4, SSE: 36175197335
İterasiya 5, SSE: 39540475405
İterasiya 6, SSE: 41134479889
İterasiya 7, SSE: 42073877408
```

Cədvəl 3.2. K-medoids alqoritmi ilə klasterləşdirmə zamanı realizə olunmuş iterasiyalar

K-medoids alqoritmi ilə klasterləşdirmə zamanı verilənlərin, klasterlərin medoidlərinin, o cümlədən, cari klasterlərə aid edilmiş verilənlərin vizullaşdırılması *Şək. 3.5*`də göstərilmişdir. Klasterlərdə yerləşdirilmiş elementlər uyğun olaraq, *sarı*, *yaşıl* və *bənövşəyi* ranglərdə, klasterlərin medoidlərini isə *qırmızı ulduz* işarələməsi ilə vizuallaşdırılmışdır.



Şək. 3.5. K-medoids alqoritmi ilə klasterləşdimənin vizuallaşdırılması

Klasterləşdirmə $T^* = 0.2914098000001104$ saniyə müddətində həyata keçirilmişdir. Proses $I_{opt} = 7$ nömrəli iterasiyada sonlandırılmış və optimal iterasiyaya uyğun $SSE = 42073877408$ qiyməti məlum olmuşdur.

```
Son mərkəzlər: [[ 3225.67192429    48.67192429    38.10094637    50.48895899]
 [ 9795.74183976    49.16023739    40.74777448    49.32047478]
 [16548.44508671    52.63294798    38.39017341    48.06647399]]
Realizə olunma müddəti: 0.17726159999983793 saniyə
```

Şək 3.6. Medoidlər və alqoritmin realizə olunma müddəti

Baza stansiyalarının optimal yerləşdirilməsi məsələsində, K-medoids alqoritmi ilə klasterləşdirmə zamanı medoidlər həqiqi əlamətlərə malik verilənlərdir. Seçilmiş medoidlərin əlamətlərinə baxaq *(Şək. 3.7.)*:

```
medoidler_cedvel = pd.DataFrame(medoidler, columns=atr)
print("Son merkezler:\n", medoidler_cedvel)
```

```
Son merkezler:
    Ehalinin_sayi  Ehalinin_sixligi  Aktiv_orta_yas  Tranzaksiya_tezliyi
0            9672                41              54                   47
1            3175                31              66                   63
2           16298                92              29                   46
```

Şək. 3.7. Medoidlərin əlamətləri

K-medoids alqoritmin baxılan məsələ üçün tapdığı optimal həll, yəni klasterlərin medoidlərini təmsil edən seçilmiş binaların adları və nömrələri *Cədvəl 3.3.*`də göstərilmişdir.

Medoidlər	Uyğun binaların adları və nömrəsi
m_1	*Lilian 171*
m_2	*Namekagon 212*
m_3	*Texas 10759*

Cədvəl 3.3. Uyğun binaların adları və nömrələri

Qeyd edək ki, K-medoids alqoritmi ilə aparılan eksperiment zamanı Python mühitində yazılmış kod fraqmentləri *Əlavələr (2)* bölməsinə əlavə olunmuşdur.

3.4. CLARA alqoritminin Python mühitində aparılmış eksperimenti

K-medoids alqoritmi ilə apardığımız eksperimentdə olduğu kimi işçi mühitə uyğun kitabxanaların qoşulması, verilənlərin ilkin emalı, həmçinin vizuallaşdırılma mərhələləri CLARA alqoritmi üçün eyni qayda ilə yerinə yetirilir.

İlk növbədə CLARA alqoritmi üçün klasterlərin, realizə olunacaq maksimal iterasiyaların, alt çoxluqların sayını və seçiləcək alt çoxluqların ölçüsünü təyin edək:

```
# Klasterlərin sayı

k = 3

# Realiə oluna biləcək maksimal iterasiyaların sayı

maks_iterasiya = 100

# Seçiləcək alt çoxluqların sayı

alt_choxluqlarin_sayi = 5

# Seçiləcək alt çoxluqların ölçüsü

alt_choxluqlarin_olchusu = 40
```

Alqoritmi realizə etmədən öncə optimal medoidləri və klasterləşdirmənin keyfiyyətini ölçən *SSE* qiymətlərini özündə saxlayan *optimal_medoidler = None* və *optimal_sse = np.inf* dəyişənlərini təyin edək. Burada *np.inf* müsbət sonsuzluq kimi təyin olunur.

```
# Optimal medoidləri ve SSE qiymətlərini özündə saxlayan dəyişənlər

optimal_medoidler = None

optimal_sse = np.inf
```

Təsadüfi yolla və təkrarlanmamaq şərti ilə alt çoxluqlar seçilir. Seçilmiş hər bir alt çoxluğa K-medoids alqoritmi tətbiq olunur və ilkin iterasiyada təsadüfi yolla medoidlər seçilir. Bu proses, *for s in range(alt_choxluqlarin_sayi):* dövrü daxilində gerçəkləşir.

```
for s in range(alt_choxluqlarin_sayi):

    # Təsadüfi yolla alt çoxluqların seçilməsi

    alt_choxluq = data[np.random.choice(data.shape[0],
alt_choxluqlarin_olchusu, replace=False)]

    # Alt çoxluqdan təsadüfi yolla medoidlərin sseçilməsi

    medoidler = alt_choxluq[np.random.choice(alt_choxluq.shape[0], k,
replace=False)]
```

Qurulmuş *for i in range(maks_iterasiya):* dövrü daxilində hər bir verilən ona məsafəcə ən yaxın olan medoid ətrafında qruplaşdırılır. Hər növbəti iterasiyada seçilmiş medoid, *for j in range(k)* dövrü boyunca hər bir verilənlə əvəz olunur. Əvəzetmə zamanı klasterlərin keyfiyyəti ölçülür və ən keyfiyyətli klasterləşdirməni doğuran verilənlər yeni medoidlər olaraq təyin olunur. Yeni medoidlərlə verilənlər arasındakı məsafələr cəminin minimum olduğu alt çoxluğun medoidləri seçilir.

```
    for i in range(maks_iterasiya):

        # Alt çoxluqdakı hər bir veriləni seçilmiş medoid ətrafında qruplaşdırın

        mesafeler = np.linalg.norm(alt_choxluq[:, np.newaxis] - medoidler,
axis=2)

        klaster_sinifleri = np.argmin(mesafeler, axis=1)
```

```
        # Medoidləri yeniləyin
        yeni_medoidler = np.array([alt_choxluq[klaster_sinifleri == j][np.argmin(mesafeler[klaster_sinifleri == j].sum(axis=1))] for j in range(k)])
        if np.all(medoidler == yeni_medoidler):
            break
        medoidler == yeni_medoidler
```

Klasterləşdirmənin keyfiyyətini ölçmək üçün *SSE (eng: sum of squared errors)* anlayışından istifadə edəcəyik və hər bir alt çoxluq üçün *SSE* qiymətini ekrana çıxaracağıq:

```
# SSE qiymətini hesabla
 sse = np.sum((data - merkezler[klaster_sinifleri]) ** 2)
# Optimal medoidleri ve SSE qiymetini yenileyin
    if sse < optimal_sse:
        optimal_medoidler = medoidler
        optimal_sse = sse
    print(f'Alt çoxluq {s+1}, SSE: {sse}')
```

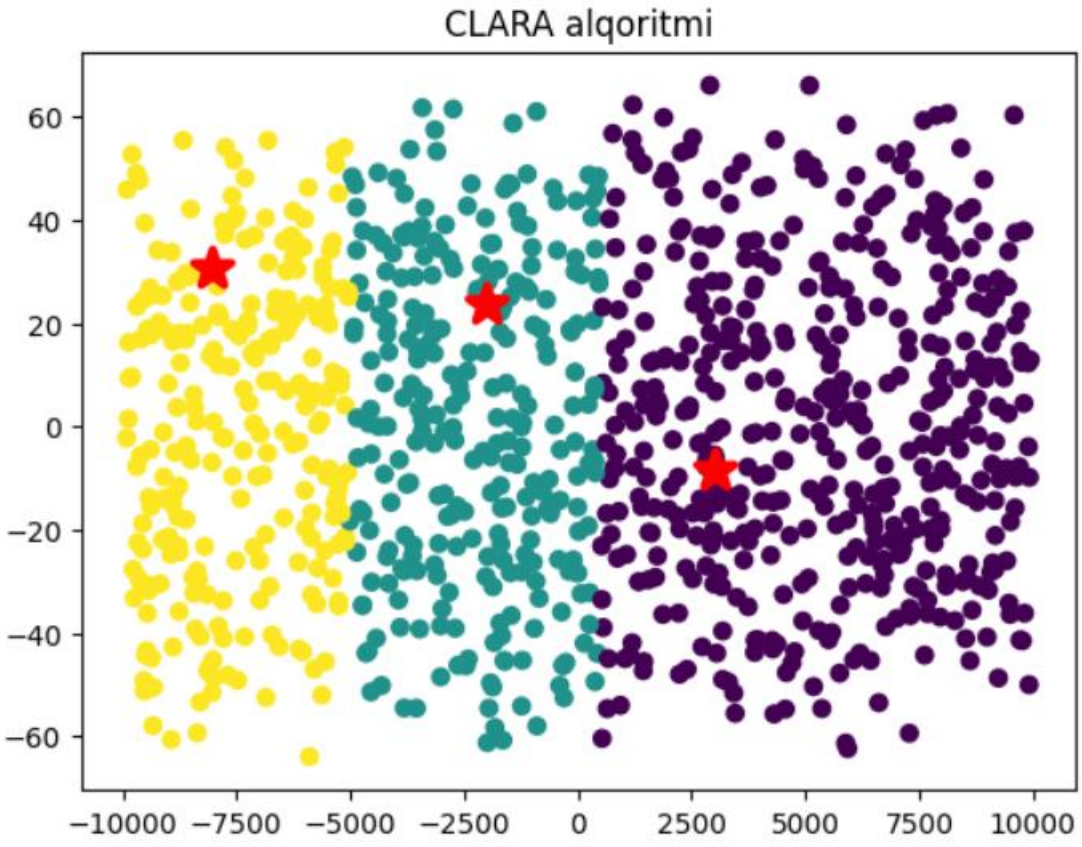
Vizuallaşdırılma mərhələsi digər alqoritmlər ilə apardığımız eksperimentdə olduğu kimi CLARA alqoritmi üçün də eyni qayda ilə yerinə yetirilir və *Sci-kit Learn* kitbaxanasının *PCA* modulundan istifadə olunur.

Eksperimentin nəticəsini analiz etmək üçün *Run* əmri ilə alqoritmi realizə etsək, alt çoxluqların hər biri üçün *SSE* qiymətini *Cədvəl 3.4.*`də görə bilərik:

Alt çoxluq 1, SSE: 7396047420
Alt çoxluq 2, SSE: 25449610256
Alt çoxluq 3, SSE: 10522854181
Alt çoxluq 4, SSE: 8653112677
Alt çoxluq 5, SSE: 7309271104

Cədvəl 3.4. Seçilmiş alt çoxluqlar üçün *SSE* qiymətlərinin təyin olunması

CLARA alqoritmi ilə klasterləşdirmə zamanı verilənlərin, klasterlərin medoidlərinin, o cümlədən, cari klasterlərə aid edilmiş verilənlərin vizullaşdırılması *Şək. 3.8*`də göstərilmişdir. Klasterlərdə yerləşdirilmiş elementlər uyğun olaraq, *sarı*, *yaşıl* və *bənövşəyi* rənglərdə, klasterlərin medoidlərini isə *qırmızı ulduz* işarələməsi ilə vizuallaşdırılmışdır.



Şək. 3.8. CLARA alqoritmi ilə klasterləşdimənin vizuallaşdırılması

Klasterləşdirmə $T^* = 0.2032112000001689$ saniyə müddətində həyata keçirilmişdir. Proses $I_{opt} = 5$ nömrəli iterasiyada sonlandırılmış və optimal iterasiyaya uyğun $SSE = 7309271104$ qiyməti məlum olmuşdur *(Şək. 3. 9.)*.

```
Son medoidler: [[ 7049    31    13    34]
 [12075    71    28    37]
 [18094    70    21    19]]
Realizə olunma müddəti: 0.2032112000001689 saniyə
SSE qiyməti: 7309271104
```

Şək. 3.9. Medoidlər və alqoritmin realizə olunma müddəti

Baza stansiyalarının optimal yerləşdirilməsi məsələsində, CLARA alqoritmi ilə klasterləşdirmə zamanı medoidlər həqiqi əlamətlərə malik verilənlərdir. Seçilmiş medoidlərin əlamətlərinə baxaq *(Şək. 3.10.)*:

```
medoidler_cedvel = pd.DataFrame(medoidler, columns=atr)
print("Son merkezler:\n", medoidler_cedvel)
```

```
Son merkezler:
    Ehalinin_sayi  Ehalinin_sixligi  Aktiv_orta_yas  Tranzaksiya_tezliyi
0           7049                31              13                   34
1          12075                71              28                   37
2          18094                70              21                   19
```

Şək. 3.10. Medoidlərin əlamətləri

CLARA alqoritminin baxılan məsələ üçün tapdığı optimal həll, yəni klasterlərin medoidlərini təmsil edən seçilmiş binaların adları və nömrələri *Cədvəl 3.5.*`də göstərilmişdir.

Medoidlər	Uyğun binaların adları və nömrəsi
m_1	*Schmedeman 36390*
m_2	*Farwell 12*
m_3	*Haas 27319*

Cədvəl 3.5. Uyğun binaların adları və nömrələri

Qeyd edək ki, CLARA alqoritmi ilə aparılan eksperiment zamanı Python mühitində yazılmış kod fraqmentləri *Əlavələr (3)* bölməsinə əlavə olunmuşdur.

3.5. CLARANS alqoritminin Python mühitində aparılmış eksperimenti

Təhlil olunmuş alqoritmlər ilə apardığımız eksperimentdə olduğu kimi işçi mühitə uyğun kitabxanaların qoşulması, verilənlərin ilkin emalı, həmçinin vizuallaşdırılma mərhələləri CLARANS alqoritmi üçün eyni qayda ilə yerinə yetirilir.

İlk növbədə CLARANS alqoritmi üçün klasterlərin, realizə olunacaq maksimal iterasiyaların, maksimal qonşu verilənlərin sayın təyin edək:

```
# Klasterlərin sayı

k = 3

# Realizə olunacaq maksimal iterasiyaların sayı

maks_iterasiya = 100

# Medoidlərə qonşu olan verilənlərin maksimal sayı

maks_qonshu = 2
```

Təsadüfi yolla k sayda medoidlər seçilir və alqoritmi realizə etmədən öncə optimal medoidləri və klasterləşdirmənin keyfiyyətini ölçən, *SSE* qiymətlərini özündə saxlayan *optimal_medoidler = None* və *optimal_sse = np.inf* dəyişənlərini təyin edək. Burada *np.inf* müsbət sonsuzluq kimi təyin olunur.

```
# Tesadufi yolla ilkin medoidleri sech

medoidler = data[np.random.choice(data.shape[0], k, replace=False)]

# SSE qiymetini ve optimal medoidleri saxlayan deyishen yarat

optimal_medoidler = medoidler.copy()

optimal_sse = np.inf
```

İterasiyalar üçün *for i in range(maks_iterasiya):* dövrü qurulur və hər bir iterasiyada medoidlərlə verilənlər arasındakı məsafələr *np.linalg.norm()* metodu ilə hesablanır və minimum məsafəni doğuran verilənlər *np.argmin()* metodu ilə eyni nömrəli klasterlərə aid edilir və klasterləşdirmənin keyfiyyəti ölçülür.

```
for i in range(maks_iterasiya):
    # Hər bir veriləni ona yaxın medoid ətrafında qruplaşdırın
    mesafeler = np.linalg.norm(data[:, np.newaxis] - medoidler, axis=2)
    klaster_sinifleri = np.argmin(mesafeler, axis=1)
    # SSE qiymətini hesablayın
    sse = np.sum((data - medoidler[klaster_sinifleri]) ** 2)
    # Optimal medoidləri ve SSE qiymətini yeniləyin
    if sse < optimal_sse:
        optimal_medoidler = medoidler.copy()
        optimal_sse = sse
```

Seçilmiş medoidlərə qonşu olan verilənlər arasında daha yaxşı həllin axtarılması zamanı *for j in range(maks_qonshu):* dövrü daxilində medoidlər və onlara qonşu olan, təsadüfi yolla seçilmiş verilənlərlə əvəz olunur.

```
    # Daha yaxşı həllin axtarışı – qonşu verilənlərin medoid kimi seçilməsi
    daha_yaxshi_hell = False
    for j in range(maks_qonshu):
```

```
    # Təsadüfi medoid və qonşu verilənin seçilməsi
    medoid_indeksi = np.random.choice(k)
    verilen_indeksi = np.random.choice(data.shape[0])
    while verilen_indeksi in medoidler:
        verilen_indeksi = np.random.choice(data.shape[0])
    # Medoidin tapılmış verilənlə əvəz olunması
    yeni_medoidler = medoidler.copy()
    yeni_medoidler[medoid_indeksi] = data[verilen_indeksi]
```

Hər bir əvəzetmə zamanı verilənlər yeni təyin olunmuş medoidlər ətrafında qruplaşdırılır və klasterləşdirmənin keyfiyyəti ölçülür. Əgər daha yaxşı həll tapılmışdırsa, medoidlər yenilənir. Əks halda, klasterlərin mərkəzləri dəyişdirilmir.

```
    # Hər bir veriləni ona yaxın medoid ətrafında qruplaşdırın
    yeni_mesafeler = np.linalg.norm(data[:, np.newaxis] - yeni_medoidler,
axis=2)
    yeni_klaster_sinifleri = np.argmin(yeni_mesafeler, axis=1)
    # Yeni SSE qiymətini hesablayın
    yeni_sse = np.sum((data - yeni_medoidler[yeni_klaster_sinifleri]) ** 2)
    # Əgər dah yaxşı həll tapılıbsa, medoidləri yenilə
    if yeni_sse < sse:
```

```
        medoidler = yeni_medoidler.copy()
        daha_yaxshi_hell = True
        break
    if not daha_yaxshi_hell:
        break
    print(f'İterasiya {i+1}')
    print(f'SSE {yeni_sse}')
```

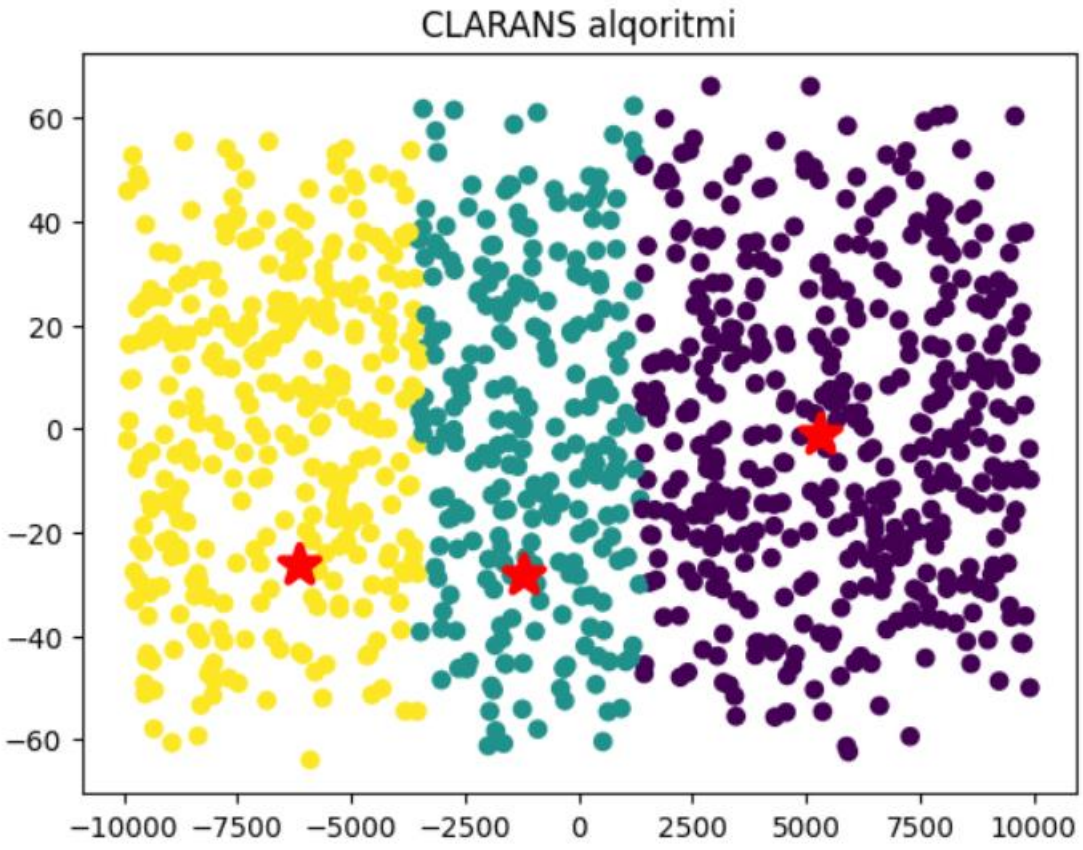
Vizuallaşdırılma mərhələsi digər alqoritmlər ilə apardığımız eksperimentdə olduğu kimi CLARANS alqoritmi üçün də eyni qayda ilə yerinə yetirilir və Matplotlib kitabxanasından, *Sci-kit Learn* kitbaxanasının *PCA* modulundan istifadə olunur.

Eksperimentin nəticəsini analiz etmək üçün *Run* əmri ilə alqoritmi realizə etsək, iterasiyaların nömrəsini və hər bir iterasiya zamanı klasterləşdirmənin keyfiyyətini *Cədvəl 3.6.* `da görə bilərik:

```
İterasiya 1, SSE 6619961642
İterasiya 2, SSE 6347502472
İterasiya 3, SSE 5988298776
İterasiya 4, SSE 5465913498
İterasiya 5, SSE 4320630434
```

Cədvəl 3.6. CLARANS alqoritmi ilə klasterləşdirmə zamanı realizə olunmuş iterasiyalar

CLARANS alqoritmi ilə klasterləşdirmə zamanı verilənlərin, klasterlərin medoidlərinin, o cümlədən, cari klasterlərə aid edilmiş verilənlərin vizullaşdırılması *Şək. 3.11*`də göstərilmişdir. Klasterlərdə yerləşdirilmiş elementlər uyğun olaraq, *sarı*, *yaşıl* və *bənövşəyi* ranglərdə, klasterlərin medoidlərini isə *qırmızı ulduz* işarələməsi ilə vizuallaşdırılmışdır.



Şək 3.11. CLARANS alqoritmi ilə klasterləşdimənin vizuallaşdırılması

Klasterləşdirmə $T^* = 0.16582830000015747$ saniyə müddətində həyata keçirilmişdir. Proses $I_{opt} = 5$ nömrəli iterasiyada sonlandırılmış və optimal iterasiyaya uyğun $SSE = 4320630434$ qiyməti məlum olmuşdur *(Şək. 3.12.)*.

```
Son medoidler: [[11285    19    55    51]
 [16186    39    62    80]
 [ 4741    62    22    74]]
Realizə olunma müddəti: 0.16582830000015747 saniyə
SSE qiymeti: 4320630434
```

Şək 3.12. Medoidlər və alqoritmin realizə olunma müddəti

Baza stansiyalarının optimal yerləşdirilməsi məsələsində, CLARANS alqoritmi ilə klasterləşdirmə zamanı medoidlər həqiqi əlamətlərə malik verilənlərdir. Seçilmiş medoidlərin əlamətlərinə baxaq *(Şək. 3.13.)*:

```
medoidler_cedvel = pd.DataFrame(medoidler, columns=atr)
print("Son merkezler:\n", medoidler_cedvel)
```

```
Son merkezler:
    Ehalinin_sayi  Ehalinin_sixligi  Aktiv_orta_yas  Tranzaksiya_tezliyi
0          11285                19              55                   51
1          16186                39              62                   80
2           4741                62              22                   74
```

Şək. 3.13. Medoidlərin əlamətləri

CLARANS alqoritminin baxılan məsələ üçün tapdığı optimal həll, yəni klasterlərin medoidlərini təmsil edən seçilmiş binaların adları və nömrələri *Cədvəl 3.5*.`də göstərilmişdir.

Medoidlər	Uyğun binaların adları və nömrəsi
m_1	*2nd 63206*
m_2	*Mandrake 55*
m_3	*Roth 3*

Cədvəl 3.7. Uyğun binaların adları və nömrələri

Qeyd edək ki, CLARANS alqoritmi ilə aparılan eksperiment zamanı Python mühitində yazılmış kod fraqmentləri *Əlavələr (4)* bölməsinə əlavə olunmuşdur.

3.6. Eksperimentlərin nəticələrinin analizi

Aparılmış eksperimentlərdə, baxılan telekommunikasiya baza stansiyalarının yerləşdirilməsi zamanı kommersial və ya yaşayış binalarının optimal seçilməsi məsələsinin dörd fərqli alqoritmin tətbiqi ilə həllinə baxılmışdır.Tədqiq olunmuş K-means alqoritmi ilə baxılan məsələnin həlli zamanı tapılmış əlamətlərə malik həqiqətən var olan binaya rast gəlinməmişdir. Məsələnin K-medoids alqoritmi ilə həlli zamanı həqiqi əlamətlərə malik binalar tapılmış və optimal yerləşdirilmənin seçilməsi üçün K-medoids alqoritminin iki fərqli modifikasiyası – CLARA və CLARANS alqoritmləri analiz olunmuş, həmin alqoritmlərin ekperimental müqayisəsinə baxılmışdır. Hər bir alqoritm üçün aparılmış ekperimentlərin nəticələri *Cədvəl 3.8.* – də öz əksini tapmışdır.

Alqoritmlərin eksperimental müqayisəsi	***Optimal həll (binanın adı və nömrəsi)***	***I_{opt} (iterasiya sayı)***	***T^* (realizə olunma müddəti)***	***SSE (klasterlərin keyfiyyət ölçüsü)***	***Klaster sayı***
K-means	*Yoxdur*	*19*	*≈0.1772*	*3736132354*	*3*
K-Medoids	*Lilian 171, Namekagon 212, Texas 10759*	*7*	*≈0.2914*	*42073877408*	*3*
CLARA	*Schemedeman 36390, Farwell 12, Haas 27319*	*5*	*≈0.2032*	*7309271104*	*3*
CLARANS	*2nd 63206 Mandrake 55 Roth 3*	*5*	*≈0.1658*	*4320630434*	*3*

Cədvəl 3.8. Aparılmış eksperimentlərin nəticəsi

Eksperimentlərin müqayisəli təhlili zamanı baxılan məsələnin həlli üçün tədqiq olunmuş K-means, K-medoids, CLARA və CLARANS alqoritmlərinin arasından ən keyfiyyətli klasterləşdirməni realizə edən və baza stansiyalarının yerləşdirilməsi məsələsinin ən optimal həllini tapan alqoritmin CLARANS alqoritmi olduğu müəyyən edilmişdir.

NƏTİCƏ

1) K-means, K-medoids, CLARA və CLARANS alqoritmləri tədqiq edilmiş, onların bir-biri ilə müqayisəli təhlili aparılmış, tətbiq sahələri araşdırılmışdır.
2) CLARA və CLARANS alqoritmlərinin Python`da proqramları tərtib edilmiş və eksperiment aparılmışdır.
3) K-means, K-medoids, CLARA və CLARANS alqoritmlərinin Python`da müqayisəli analizi aparılmışdır. Eksperimentin nəticələrinin analizi zamanl aydın olur ki, tədqiq olunmuş alqoritmlərin arasından ən yaxşı nəticə CLARANS alqoritmi ilə əldə olunmuşdur. Alqoritmlərin müqayisəsi zamanı effektivlik göstəricisi kimi *SSE* qiymətlərindən istifadə olunmuşdur.

İSTİFADƏ EDİLMİŞ ƏDƏBİYYAT

1. Ramiz Alıquliyev, Şəlalə Tahirzadə. Böyük həcmli fərdi məlumatların analizi üçün iterativ çəkili k-means alqoritmi. İnformasiya təhlükəsizliyinin aktual multidissiplinar elmi-praktiki problemləri. Noyabr 2019.
2. Dataqoil. K-medoids Clustering from Scratch in Python. Feb 2022.
3. Datanovia. CLARA in R: Clustering Large Applications.2018
4. Duke University. Mark Ricahrdson. Lecture - Principial Component Analysis. May 2009
5. Daniel Schmidt, Mohammed Ibrahim, Sven Lautenschlager. Hochschule Wismar – University of Applied Sciences. CLARANS. Germany 2013.
6. E.M.Mirkes. University of Leicester. Lecture - K-medoids and K-means standart algorithm. 2011.
7. E.Mahima Jane, E.George Dharma Prakash Raj.Survey on Partition based Clustering Algorithms in Big Data. International Journal of Computer Science and Engineering. Dec 2017
8. Gopi Gandhi, Rohit Srivatava. Analysis And Implementation of Modified K-Medoids Algorithm to Increase Scalability And Efficiency for Large Dataset. International Journal of Research Engineering and Technology. Jun 14, 2014.
9. Javatpoint. Modifications of K-medoids: CLARA and CLARANS algorithm.
10. Jiawei Han, Micheline Kamber, Jian Pei. Data Mining: Concepts and Techniques. Third edition.USA.2012.
11. Mo Tiwari, Martin Jinye Zhang, James Mayclin, Sebastian Thrun, Chris Piech, Ilan Shomorony. BanditPAM: Almost Linear Time k-Medoids Clustering via Multi-Armed Bandits.December 2020.
12. Pandas – Python library documentation and techniques.
13. Pranjal Awasthi, Afonso Bandeira, Moses Charikar. Technische Universität Berlin. Some mathematics for k-means clustering. Berlin. December 2015

14. Raymond T. NG, Jiawei Han. CLARANS: A Method for Clustering Objects for Spatial Data Mining. IEEE Transactions on Knowledge and Data Engineering.Vol 14.No 5.2022, pp. 1003 – 1016.

15. Saket Thavanni. K-means Clustering: Optimizing Cost Function Mathematically. Medium.April 28, 2020.

16. Swarndeep Saket J, Dr. Sharnil Pandya. An Overview of Partitioning Algotihms in Clustering Techniques. IJARCET June 2016.

17. University of California San Diego. CSE 291.Lecture 2 – The K-means Clustering Problem. Spring 2008.

18. University of California San Diego.CSE 291.Lecture 3 – The K-medoids Clustering Problem. Spring 2008.

19. Wikipedia. Principial Compenent Analysis. ENG

20. Wikipedia. Lloyd`s algorithm.ENG

21. YanPing Zhao, XiaoLai Zhou. K-means Clustering and Its Improvement Research.Journal of Physics: Conference Series. China. IWECAI 2021

ƏLAVƏLƏR

1) **K-means alqoritmi**nin Python mühitində aparılmış eksperimentində yazılmış kod fraqmentləri:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from timeit import default_timer as timer

start = timer()

# Verilenlerin muhite elave olunmasi
data = pd.read_csv('station.csv')
atr = ["Ehalinin_sayi",
"Ehalinin_sixligi","Aktiv_orta_yas","Tranzaksiya_tezliyi"]
data = data[atr].copy()

# Tabular data tipli verilenlerin massiv formasina getirilmesi
data = data.to_numpy()

# Klasterlerin sayi
k = 3

# Realize olunacaq maksimum iterasiyalarin sayi
maks_iterasiya = 100

# Tesadufi yolla merkezlerin sechilmesi
merkezler = data[np.random.choice(data.shape[0], k, replace=False)]
```

```
for i in range(maks_iterasiya):
    # Her bir verileni ona yaxin merkez etrafinda qruplashdir
    mesafeler= np.linalg.norm(data[:, np.newaxis] - merkezler, axis=2)
    klaster_sinifleri = np.argmin(mesafeler, axis=1)

    # Merkezleri yenile
    yeni_merkezler = np.array([data[klaster_sinifleri == j].mean(axis=0) for j in
range(k)])
    if np.all(merkezler == yeni_merkezler):
        break
    merkezler = yeni_merkezler

    # SSE qiymetini hesabla
    sse = np.sum((data - merkezler[klaster_sinifleri]) ** 2)
    print(f'İterasiya {i+1}, SSE: {sse}')

# PCA ile vizuallashdir
pca = PCA(n_components=2)
data_pca = pca.fit_transform(data)
merkezler_pca = pca.transform(merkezler)

plt.scatter(data_pca[:, 0], data_pca[:, 1], c=klaster_sinifleri)
plt.scatter(merkezler_pca[:, 0], merkezler_pca[:, 1], marker='*', s=200,
linewidths=3,
            color='r', zorder=10)
plt.title('K-means alqoritmi')
plt.show()
```

```
# Son merkezleri ekrana chixart
print('Son mərkəzlər:', merkezler)
end = timer()
print(f"Realizə olunma müddəti: {end - start} saniyə")
```

2) **K-medoids alqoritmi**nin Python mühitində aparılmış eksperimentində yazılmış kod fraqmentləri:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from timeit import default_timer as timer

start = timer()

# Verilenlerin muhite elave olunmasi
data = pd.read_csv('station.csv')
atr = ["Ehalinin_sayi",
"Ehalinin_sixligi","Aktiv_orta_yas","Tranzaksiya_tezliyi"]
data = data[atr].copy()

# Tabular data tipli verilenlerin massiv formasina getirilmesi
data = data.to_numpy()

# Klasterlerin sayi
k = 3

# Realize olunacaq maksimum iterasiya sayi
maks_iterasiya = 100
```

```
# Tesadufi yolla medoidlerin sechilmesi
medoidler = data[np.random.choice(data.shape[0], k, replace=False)]

for i in range(maks_iterasiya):
    # Her bir verileni ona yaxin merkez etrafinda qruplashdir
    mesafeler = np.linalg.norm(data[:, np.newaxis] - medoidler, axis=2)
    klaster_sinifleri = np.argmin(mesafeler, axis=1)

    # Medoidleri yenile
    yeni_medoidler = np.array([data[klaster_sinifleri ==
j][np.argmin(mesafeler[klaster_sinifleri == j].sum(axis=1))] for j in range(k)])
    if np.all(medoidler == yeni_medoidler):
        break
    medoidler = yeni_medoidler

    # SSE qiymetini hesabla
    sse = np.sum((data - medoidler[klaster_sinifleri]) ** 2)
    print(f'İterasiya {i+1}, SSE: {sse}')

# PCA ile vizuallashdir
pca = PCA(n_components=2)
data_pca = pca.fit_transform(data)
medoidler_pca = pca.transform(medoidler)

plt.scatter(data_pca[:, 0], data_pca[:, 1], c=labels)
plt.scatter(medoidler_pca[:, 0], medoidler_pca[:, 1], marker='*', s=200,
linewidths=3,
            color='r', zorder=10)
```

```
plt.title('K-medoids alqoritmi')
plt.show()

# Son medoidleri ekranda goster
print('Son medoidler:', medoidler)
end = timer()
print(f"Realizə olunma müddəti: {end - start} saniyə")
```

3) **CLARA alqoritmi**nin Python mühitində aparılmış eksperimentində yazılmış kod fraqmentləri:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from timeit import default_timer as timer

start = timer()

# Verilenlerin muhite elave olunmasi
data = pd.read_csv('station.csv')
atr=["Ehalinin_sayi","Ehalinin_sixligi","Aktiv_orta_yas","Tranzaksiya_tezliyi"]
data = data[atr].copy()

# Tabular data tipli verilenlerin massiv formasina getirilmesi
data = data.to_numpy()

# Klasterlerin sayi
k = 3
```

```
# Realize oluna bilecek maksimal iterasiyalarin sayi
maks_iterasiya = 100

# Sechilecek alt choxluqlarin sayi
alt_choxluqlarin_sayi = 5

# Sechilecek alt choxluqlarin olchusu
alt_choxluqlarin_olchusu = 40

# Optimal medoidleri ve SSE qiymetlerini ozunde saxlayan deyishenlerin teyini
optimal_medoidler = None
optimal_sse = np.inf

for s in range(alt_choxluqlarin_sayi):
    # Tesadufi yolla verilenlerden alt choxluqlarin sechilmesi
    alt_choxluq = data[np.random.choice(data.shape[0], alt_choxluqlarin_olchusu, replace=False)]

    # Alt choxluqdan tesadufi yolla medoidlerin sechilmesi
    medoidler = alt_choxluq[np.random.choice(alt_choxluq.shape[0], k, replace=False)]

    for i in range(maks_iterasiya):
        # Alt choxluqdaki her bir verileni sechilmish medoid etrafinda qruplashdirin
        mesafeler = np.linalg.norm(alt_choxluq[:, np.newaxis] - medoidler, axis=2)
        klaster_sinifleri = np.argmin(mesafeler, axis=1)

        # Medoidleri yenileyin
```

```
        yeni_medoidler = np.array([alt_choxluq[klaster_sinifleri == j][np.argmin(mesafeler[klaster_sinifleri == j].sum(axis=1))] for j in range(k)])
        if np.all(medoidler == yeni_medoidler):
            break
        medoidler == yeni_medoidler

    # Verilenlerin her birini ona yaxin medoidler etrafinda qruplashdirin
    mesafeler = np.linalg.norm(data[:, np.newaxis] - medoidler, axis=2)
    klaster_sinifleri = np.argmin(mesafeler, axis=1)

    # Umumi SSE qiymetini hesablayin
    sse = np.sum((data - medoidler[klaster_sinifleri]) ** 2)

    # Optimal medoidleri ve SSE qiymetini yenileyin
    if sse < optimal_sse:
        optimal_medoidler = medoidler
        optimal_sse = sse

    print(f'Alt çoxluq {s+1}, SSE: {sse}')

# Verilenlerin her birini ona yaxin medoidler etrafinda qruplashdirin
mesafeler = np.linalg.norm(data[:, np.newaxis] - optimal_medoidler, axis=2)
klaster_sinifleri = np.argmin(mesafeler, axis=1)

# PCA ile vizuallashdir
pca = PCA(n_components=2)
data_pca = pca.fit_transform(data)
medoidler_pca = pca.transform(optimal_medoidler)
```

```
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=klaster_sinifleri)
plt.scatter(medoidler_pca[:, 0], medoidler_pca[:, 1], marker='*', s=200,
linewidths=3,
        color='r', zorder=10)
plt.title('CLARA alqoritmi')
plt.show()

# Son medoidleri ekranda goster
print('Son medoidler:', optimal_medoidler)
end = timer()
print(f"Realizə olunma müddəti: {end - start} saniyə")
print('SSE qiyməti:', optimal_sse)
print('Optimal iterasiya sayı:', i+1)
```

4) **CLARANS alqoritmi**nin Python mühitində aparılmış eksperimentində yazılmış kod fraqmentləri:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from timeit import default_timer as timer

start = timer()

# Verilenlerin muhite elave olunmasi
data = pd.read_csv('station.csv')
atr=["Ehalinin_sayi",
"Ehalinin_sixligi","Aktiv_orta_yas","Tranzaksiya_tezliyi"]
```

```
data = data[atr].copy()

# Tabular data tipli verilenlerin massiv formasina getirilmesi
data = data.to_numpy()

# Klasterlerin sayi
k = 3

# Realize olunacaq maksimum iterasiyalarin sayi
maks_iterasiya = 100

# Maksimal qonshu verilenlerin sayi
maks_qonshu = 2

# Tesadufi yolla ilkin medoidleri sech
medoidler = data[np.random.choice(data.shape[0], k, replace=False)]

# SSE qiymetini ve optimal medoidleri saxlayan deyishen yarat
optimal_medoidler = medoidler.copy()
optimal_sse = np.inf

for i in range(maks_iterasiya):
    # Her bir verileni ona yaxin medoid etrafinda qruplashdir
    mesafeler = np.linalg.norm(data[:, np.newaxis] - medoidler, axis=2)
    klaster_sinifleri = np.argmin(mesafeler, axis=1)

    # SSE qiymetini hesabla
    sse = np.sum((data - medoidler[klaster_sinifleri]) ** 2)
```

```
    # Optimal medoidleri ve SSE qiymetini yenile
    if sse < optimal_sse:
        optimal_medoidler = medoidler.copy()
        optimal_sse = sse
    # Daha yaxshi hell uchun axtarish edilmesi - diger verilenlerin medoid kimi sechilmesi
    daha_yaxshi_hell = False
    for j in range(maks_qonshu):
        # Tesadufi medoid ve verilen sechin
        medoid_indeksi = np.random.choice(k)
        verilen_indeksi = np.random.choice(data.shape[0])
        while verilen_indeksi in medoidler:
            verilen_indeksi = np.random.choice(data.shape[0])

        # Medoidi tapilmish verilenle evez ele
        yeni_medoidler = medoidler.copy()
        yeni_medoidler[medoid_indeksi] = data[verilen_indeksi]

        # Her bir verileni ona yaxin medoid etrafinda qruplashdir
        yeni_mesafeler = np.linalg.norm(data[:, np.newaxis] - yeni_medoidler, axis=2)
        yeni_klaster_sinifleri = np.argmin(yeni_mesafeler, axis=1)

        # Yeni SSE qiymetini hesabla
        yeni_sse = np.sum((data - yeni_medoidler[yeni_klaster_sinifleri]) ** 2)

        # Eger daha yaxhsi hell tapilibsa, medoidleri yenile
        if yeni_sse < sse:
            medoidler = yeni_medoidler.copy()
```

```
            daha_yaxshi_hell = True
            break

    if not daha_yaxshi_hell:
        break

    print(f'İterasiya {i+1}')
    print(f'SSE {yeni_sse}')

# Her bir verileni ona yaxin medoid etrafinda qruplashdir
mesafeler = np.linalg.norm(data[:, np.newaxis] - optimal_medoidler, axis=2)
klaster_sinifleri = np.argmin(mesafeler, axis=1)

# PCA istifade ederek vizuallashdir
pca = PCA(n_components=2)
data_pca = pca.fit_transform(data)
medoidler_pca = pca.transform(optimal_medoidler)

plt.scatter(data_pca[:, 0], data_pca[:, 1], c=sinifler)
plt.scatter(medoidler_pca[:, 0], medoidler_pca[:, 1], marker='*', s=200,
linewidths=3,
            color='r', zorder=10)
plt.title('CLARANS alqoritmi')
plt.show()

# Son medoidleri ekranda goster
print('Son medoidler:', optimal_medoidler)
end = timer()
print(f"Realizə olunma müddəti: {end - start} saniyə")
```

```
print('SSE qiymeti:', optimal_sse)
print('Optimal iterasiya sayı:', i+1)
```