

Ministerul Educației și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

**RAPORT**

**Lucrarea de laborator nr.3**

**la Inteligența Artificială**

*Tema: Utilizarea diferitor algoritmi de clusterizare pe diferite tipuri de date. Vizualizarea datelor.*

Grupa academică: TI-214  
A efectuat: Reguș Ruslan

**A verificat: Mariana Rusu**

Chișinău 2024

**Problema propusă:**

Alegeți spre implementare (în limbajul Python) câte o metodă din fiecare categorie de mai sus (cel puțin patru algoritmi). Explicați conceptul și pașii de bază ai fiecărui algoritm. Alegeți sau creați un set de date conform necesităților fiecărui algoritm elaborat. Evaluați rezultatele obținute.

**Tipuri de clustering**

Crisp vs fuzzy clustering

* Crisp clustering = fiecare dată aparţine unui singur cluster.
* Fuzzy clustering = o dată poate aparţine mai multor clustere (grad de apartenenţă pentru fiecare cluster).

Flat vs hierarchical clustering

* Flat (partitional) clustering = rezultatul este un set de clustere (o partiţie).
* Hierarchical clustering = rezultatul este o ierarhie de partiţii.

Variante de algoritmi

* Algoritmi partiţionali (ex: kMeans, Fuzzy cMeans).
* Algoritmi hierarhici (alg. aglomerativi, alg. divizivi).
* Algoritmi bazaţi pe densitate (ex: DBSCAN).
* Algoritmi bazați pe modele probabiliste (ex: EM = Expectation Maximization)

Clusterizarea este o metodă de învățare nesupravegheată în care se extrag referințe din seturi de date constând în date de intrare fără răspunsuri etichetate. Aceasta este sarcina de a împărți punctele de date într-un număr de grupuri astfel încât punctele de date din aceleași grupuri să fie mai asemănătoare cu alte puncte de date din același grup și să fie diferite de punctele de date din alte grupuri. În esență, este vorba de o colectare a obiectelor pe baza similitudinii și a disimilarității dintre ele. În cadrul laboratorului se vor explora diverși algoritmi, precum kMeans, DBSCAN, hierarhici aglomerativi, dar și expactation maximization.

**Algoritmi partiționali: kMeans**

Algoritmul k-Means este o tehnică populară de învățare automată nesupravegheată utilizată pentru clusterizarea datelor. Acesta ajută la gruparea punctelor de date similare, fără a avea cunoștințe prealabile despre etichetele sau categoriile datelor.

Câteva aplicații comune ale clusterizării k-means se pot identifica după cum urmează:

* Segmentarea clienților: Gruparea clienților în funcție de istoricul lor de cumpărături sau date demografice pentru a dezvolta campanii de marketing țintite.
* Segmentarea imaginilor: Identificarea obiectelor sau regiunilor distincte din cadrul unei imagini, cum ar fi separarea prim-planului de fundal.
* Detectarea anomaliilor: Detectarea punctelor de date care deviază semnificativ de la majoritate, indicând potențial valori aberante sau erori.
* Sisteme de recomandare: Recomandarea de produse sau servicii utilizatorilor pe baza preferințelor lor anterioare și a comportamentului utilizatorilor similari dintr-un cluster.

În cadrul acestui algoritm se va folosi datele privind locuințele din California prezent în setul de date de pe Kaggle. Astfel, se va folosi date de localizare (latitudine și longitudine), precum și valoarea mediană a casei. Apoi se va grupa casele în funcție de locație și se va observa cum fluctuează prețurile caselor în California.

Pentru primul pas se va salva setul de date ca un fișier csv numit "housing.csv" în directorul de lucru și se va citi conform reprezentării figurii 1.

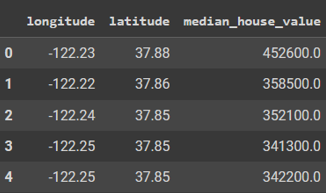


Figura 1 – Salvare set de date

Setul de date utilizat include 3 variabile selectate cu ajutorul parametrului usecols:

* Longitudine: O valoare care indică distanța unei case la vest. Valorile mai mari corespund caselor situate mai spre vest.
* Latitudine: O valoare care indică distanța unei case la nord. Valorile mai mari corespund caselor situate mai spre nord.
* Valoarea mediană a casei: Prețul median al casei în cadrul unui bloc, exprimat în USD.

În continuare se va vizualiza datele privind locuințele, așa încât se va analiza distribuția spațială a prețurilor medii ale caselor prin intermediul unei hărți termice, aceasta fiind afișată în figura 1.1

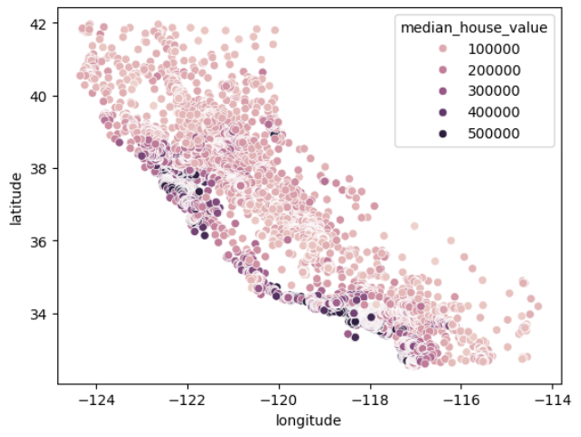


Figura 1.1 – Vizualizare date locuințe

Algoritmii bazați pe distanțe, precum Clusterizarea k-Means, necesită normalizarea datelor pentru a le aduce la o scară comparabilă. Omiterea normalizării poate duce la distorsiuni în rezultate, favorizând variabilele cu scale mai mari. De exemplu, includerea prețului caselor (cu o scară semnificativ mai mare) alături de latitudine și longitudine (delimitate) ar conferi prețului o influență disproporționată în optimizare. Inițial, s-a configurat diviziunile de instruire și de testare folosind train\_test\_split din sklearn, așa încât s-a alocat 33% din datele totale setului de testare, iar restul de 67% vor fi folosite pentru antrenarea modelului.

În continuare, se vor normaliza datele de instruire și de testare utilizând metoda preprocessing.normalize() din sklearn, aceasta fiind ilustrată în figura 1.2.

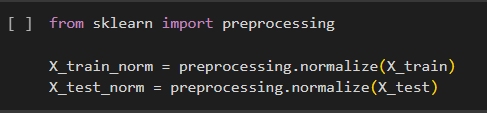


Figura 1.2 – Normalizarea datelor

În prima iterație, se alege arbitrar numărul de clustere (k) ca fiind 3. Apoi, se creează o instanță a KMeans, se definește numărul de clustere cu atributul n\_clusters, se setează n\_init la "auto" pentru a specifica numărul de iterații cu diferite semințe de centroizi, și se stabilește random\_state la 0 pentru consistență în rezultate. Modelul este apoi ajustat la datele de instruire normalizate prin metoda fit(), acestea fiind illustrate în figura 1.3.



Figura 1.3 – Ajustare model

Odată ce datele sunt adaptate, se accesa etichetele din atributul labels\_, iar reprezentarea acestora pot fi vizualizate în figura 1.4.

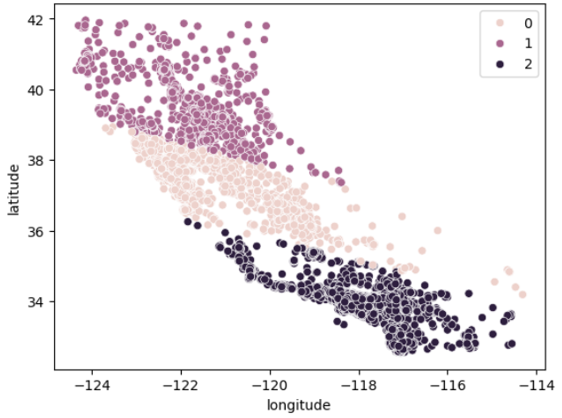


Figura 1.4 – Reprezentare date

Se observă că datele sunt acum clar separate în 3 grupuri distincte, corespunzând regiunilor California de Nord, California Centrală și California de Sud. De asemenea, se poate analiza distribuția prețurilor medii ale locuințelor în aceste 3 grupuri folosind un boxplot, acesta fiind reprezentat în figura 1.5

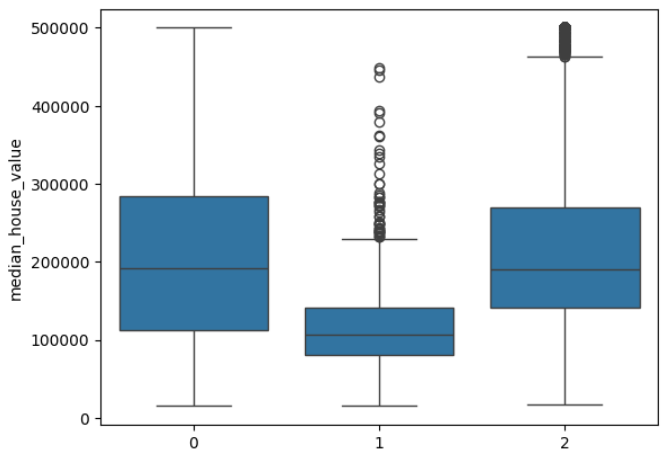


Figura 1.5 – Distribuție prețuri medii ale locuințelor

În continuare, se poate evalua performanța algoritmului de grupare folosind un scor Silhouette, care face parte din sklearn.metrics, unde un scor mai mic reprezintă o potrivire mai bună, acesta fiind reprezentat în figura 1.6.



Figura 1.6 – Evaluare performanță model

Un dezavantaj semnificativ al algoritmului K-Means constă în dificultatea stabilirii numărului optim de clustere (k). Algoritmul nu oferă indicii directe cu privire la valoarea ideală a lui k, prin urmare se vor testa mai multe opțiuni ale numărului de clustere, precum k=2, 4, 5. De obicei, creșterea valorii lui K îmbunătățește performanța clustering-ului până la un punct, după care observăm randamente descrescătoare sau performanțe mai slabe. În cazul în care valoarea lui k=5, rezultatele pot fi vizualizate în figura 1.7.

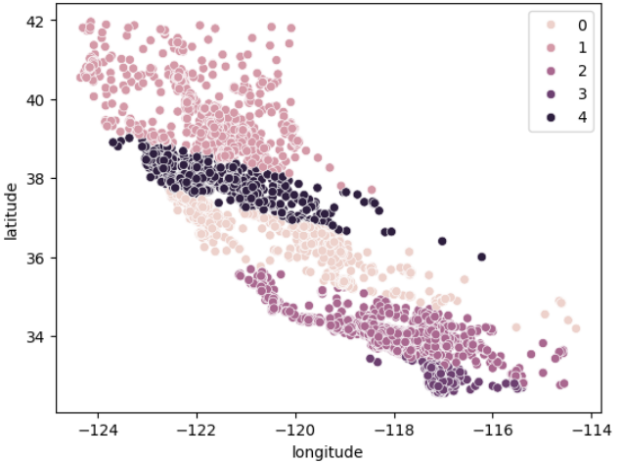


Figura 1.7 – Reprezentare rezultate k=5

Conform diagramei din figura 1.8 se observă că valoarea optimă a lui k pare a fi 5, fiind probabil cel mai bun rezultat fără a cădea în supraajustare.

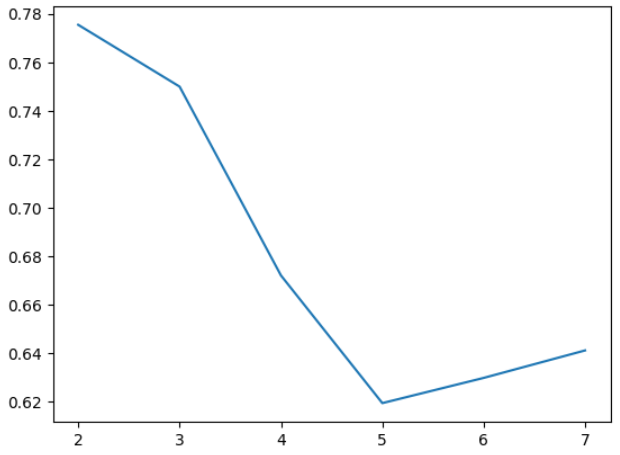


Figura 1.8 – Reprezentare rezultate model antrenat

**Algoritmi hierarhici aglomerativi.**

O abordare ierarhică de grupare se bazează pe determinarea de grupe succesive pe baza unor grupe definite anterior. Este o tehnică care vizează mai mult gruparea datelor într-un arbore de clustere numit dendrogramă, care reprezintă grafic relația ierarhică dintre clusterele de bază.

Clusteringul ierarhic, în special varianta sa aglomerativă, își găsește numeroase aplicații în diverse domenii, datorită capacității sale de a descoperi structuri ierarhice în date:

* Analiza imaginilor: Segmentarea imaginilor pentru a identifica obiecte (de ex., mașini, oameni, animale) și regiuni distincte.
* Analiza datelor financiare: Gruparea clienților în funcție de comportamentul lor financiar (cheltuieli, investiții, risc) pentru a dezvolta strategii de marketing personalizate.
* Analiza textului: Gruparea documentelor sau articolelor științifice în funcție de subiect pentru a facilita navigarea și accesul la informație.

Primul scenariu implică abordarea explicată mai sus, începând cu considerarea fiecărei observații ca un cluster singular (cu un singur punct de date). Apoi, se realizează unirea iterativă a clusterelor până când se obține un singur cluster, într-un proces cunoscut și sub numele de abordare de jos în sus. În ilustrația din figura 2 se va considera următoarele:

* Se începe prin considerarea fiecărui animal ca un cluster unic.
* Se generează apoi trei cluster-uri diferite în funcție de similaritățile lor: Păsări (Vultur și Păun), Mamifere (Leu și Urs), Animale cu mai mult de trei picioare (Păianjen și Scorpion).
* Procesul de fuziune se repetă pentru a crea cluster-ul de vertebrate, combinând cele două cluster-e cele mai similare: Păsări și Mamifere.
* După această etapă, cele două cluster-e rămase, Vertebrate și Animale cu mai mult de trei picioare, sunt unite pentru a crea un singur cluster de Animale.

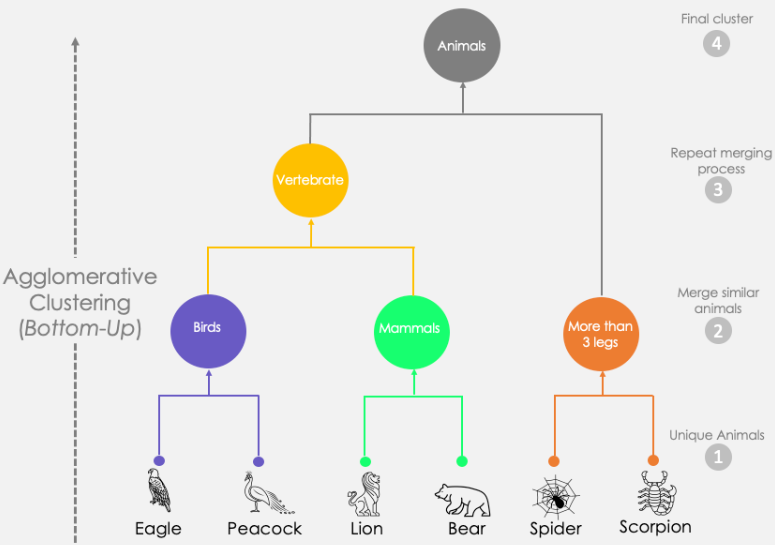


Figura 2 – Pași de implementare algoritm hierarhic aglomerativ

În continuare, se va importa modulele necesare și se va încărca setul de date ce conține 9 500 de împrumuturi cu informații despre structura împrumutului, împrumutat și dacă împrumutul a fost rambursat integral sau nu, acesta fiind vizualizat în figura 2.1.

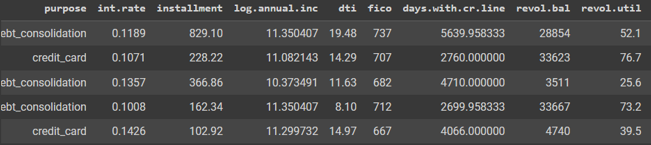


Figura 2.1 – Vizualizare set de date

Înainte de aplicarea grupării, datele trebuie să fie preprocesate pentru a trata informațiile lipsă, pentru a normaliza valorile coloanelor și pentru a elimina coloanele irelevante. Apoi, se va analiza datele privind împrumuturile, utilizând toate coloanele, cu excepția celor două, și anume:

* Coloana "Scop"
* Coloana "not.fully.paid", deoarece aceasta reprezintă eticheta care indică dacă împrumutatul a efectuat sau nu o plată integrală.

Datele curățate (cleaned\_data), reprezentate în figura 2.2, corespund informațiilor fără aceste două coloane menționate anterior.

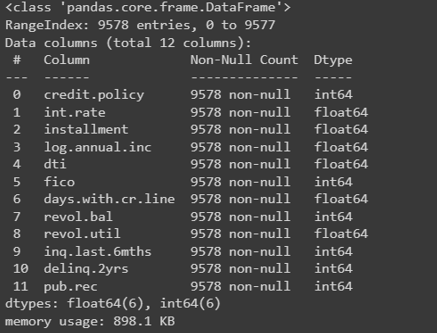


Figura 2.2 – Date curățate

Unul dintre punctele slabe ale grupării ierarhice este faptul că este sensibilă la valorile aberante, iar distribuția fiecărei variabile este dată de boxplot, după cum urmează în figura 2.3.

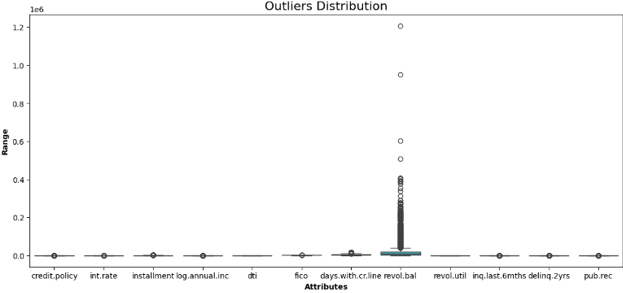


Figura 2.3 – Distribuție variabile

Prin utilizarea abordării intervalului interquartil, se poate elimina toate punctele care se află în afara intervalului definit de cuartilele +/-1,5 \* IQR, unde IQR este intervalul interquartil. După modificările efectuate și vizualizarea datelor curățate, s-a identificat care este forma datelor, respectiv 9.319 rânduri și 12 coloane. Acest lucru presupune că 259 de observații erau aberante și au fost eliminate.

Un alt pas important este redimensionarea datelor înainte de gruparea ierarhică, deoarece aceasta utilizează distanța euclidiană, fiind sensibilă la variabilele cu scări diferite. Pentru acest scop, se utilizează clasa StandardScaler din biblioteca sklearn, aceasta fiind reprezentată în figura 2.4.



Figura 2.4 – Redimensionarea datelor

În continuare, se va specifica metoda de clusterizare în atributul linkage(), iar drept implementare, vor fi explorate toate cele trei tehnici de legătură, utilizând distanța euclidiană, după cum este reprezentat în figura 2.5.



Figura 2.5 – Specificare metode de clusterizare

După calcularea celor trei grupări, dendrogramele corespunzătoare sunt prezentate în continuare, începând cu gruparea completă, fiind reprezentată în figura 2.6.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

Figura 2.6 – Reprezentare diagrama grupare completă

În continuare, funcția dendrogram afișează o reprezentare vizuală a clusterelor ierarhice sub forma unei dendrograme, respectiv va realiza un clustering ierarhic utilizând metoda "average" aceasta fiind afișată în figura 2.7.

|  |  |
| --- | --- |
|  |  |

Figura 2.7 – Reprezentare diagramă de tip average\_clustering

Apoi, se va vizualiza funcția dendrogram utilizând metoda ”single”, aceasta fiind afișată în figura 2.8

|  |  |
| --- | --- |
|  |  |

Figura 2.8 – Reprezentare diagrama de tip single\_clustering

Pentru fiecare metodă de clusterizare, se analizează modul în care dendrograma este construită și cum fiecare punct de date este atribuit într-un singur cluster. Pe axa x a dendrogramei sunt reprezentate eșantioanele din date, în timp ce axa y indică distanța dintre aceste eșantioane. Cu cât linia este mai înaltă, cu atât sunt mai diferite acele eșantioane sau clustere. Numărul corespunzător de clustere se obține prin trasarea unei linii orizontale prin cea mai înaltă linie verticală, iar numărul de intersecții indică numărul de clustere.

Determinarea numărului optim de clustere se face identificând linia verticală cea mai înaltă care nu se intersectează cu niciun alt cluster (linie orizontală). Aceasta este evidențiată printr-un cerc roșu și o bifă verde în figurile 2.6, 2.7 și 2.8

Respectiv, după găsirea numărului optim de clustere, se va reprezenta ce înseamnă aceste clustere în funcție de scorul de credit al împrumutatului, acestea fiind reprezentate în figura 2.9.

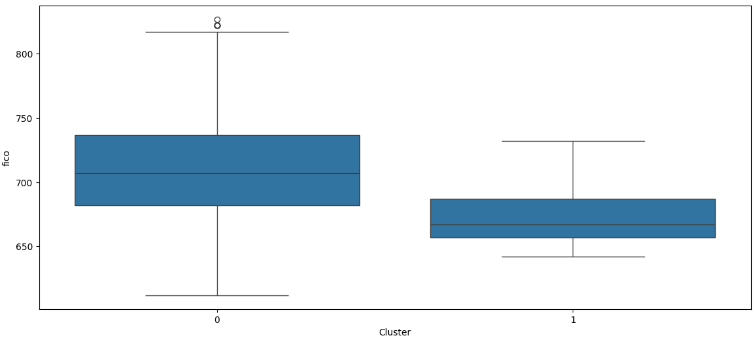


Figura 2.9 – Reprezentare bloxplot

Respectiv, conform boxplot-ului de mai sus, se poate observa că:

* Împrumutații din clusterul 0 au cele mai mari scoruri de credit, în timp ce debitorii din clusterul 1 au scoruri de credit mai mici.

**Algoritmi bazaţi pe densitate: DBSCAN**

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) este un algoritm de clusterizare ierarhic care utilizează densitatea punctelor de date pentru a le grupa. Spre deosebire de algoritmii bazati pe distanță, DBSCAN nu necesită predefinirea numărului de clustere.

Principiul de funcționare:

* Algoritmul definește doi parametri: ε (epsilon) și MinPts (minimum points).
* Un punct este considerat "core" dacă are cel puțin MinPts puncte în vecinătatea sa definită de ε.
* Punctele din vecinătatea unui punct core sunt considerate parte a aceluiași cluster.
* Punctele care nu sunt core și nu pot fi conectate la un cluster existent sunt considerate "zgomot".

Domenii de aplicabilitate:

* Analiza imaginilor: Segmentarea imaginilor pentru a identifica obiecte.
* Analiza datelor financiare: Gruparea clienților în funcție de comportamentul lor financiar.
* Bioinformatică: Gruparea genelor cu funcții biologice similare.
* Detectarea fraudei: Identificarea tranzacțiilor frauduloase.

Pentru acest algoritm se va utiliza datele de segmentare a clienților din Mall. Acestea conțin vârsta, sexul, venitul și scorul de cheltuieli al clienților. Respectiv, se va folosi aceste caracteristici pentru a crea diverse clustere. În primul rând, se va încărca setul de date, apoi, vom selecta trei coloane ("Vârsta", "Venitul anual (k$)", "Scorul cheltuielilor (1-100)") pentru a crea cadrul de date X\_train, acesta fiind reprezentat în figura 3.

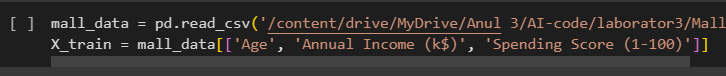


Figura 3 – Creare cadru de date X\_train

Apoi se va ajusta X\_train pe algoritmul DBSCAN cu eps 12,5 și min\_sample 4, acestea fiind reprezentată în figura 3.1

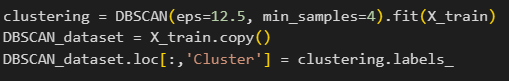


Figura 3.1 – Ajustare valori X\_train

În continuare, pentru a vizualiza distribuția clusterelor, se utilizează funcția value\_counts() și se convertește într-un cadru de date. Respectiv, în figura 3.2, se observă că există 5 clustere și 1 valoare aberantă. Clusterul `0` este cel mai mare, cu 112 rânduri.

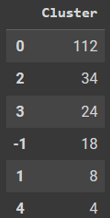


Figura 3.2 – Reprezentare distribuție clustere

Spre final, se va realiza o digramă, în figura 3.3, în care se reprezintă în mod clar cum fiecare client face parte din unul dintre cele 5 grupuri, iar noi putem folosi aceste informații pentru a oferi oferte de top clienților cu grupuri mov și oferte mai ieftine clienților cu grupuri verde închis.

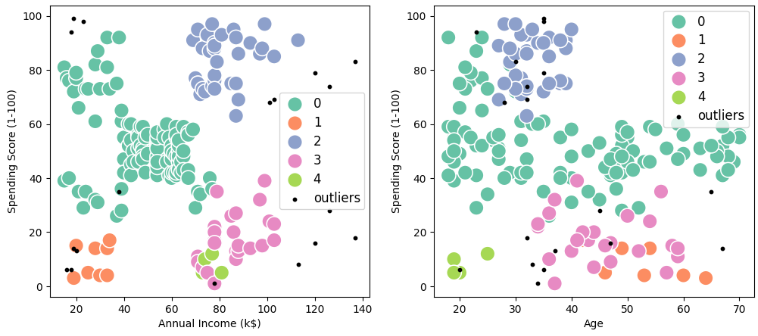


Figura 3.3 – Repartizare valori

**Algoritmi bazați pe modele probabiliste: Expectation Maximization**

Algoritmul Expectation Maximization (EM) este o metodă iterativă utilizată pentru a estima parametrii unui model probabilistic din date incomplete sau cu zgomot. Algoritmul EM este aplicabil variabilelor latente, care sunt variabile care nu sunt direct observabile, dar sunt deduse din valorile altor variabile observate.

Principiul de funcționare, a cărui schemă bloc este reprezentat în figura 4, este după cum urmează:

* Etapa de așteptare (E): Se calculează distribuția probabilistă a variabilelor ascunse, condiționată de datele observate și parametrii modelului actuali.
* Etapa de maximizare (M): Se estimează parametrii modelului care maximizează probabilitatea datelor observate, ținând cont de distribuția variabilelor ascunse calculată în etapa E.
* Repetarea: Etapele E și M se repetă până când se obține convergență, adică parametrii modelului nu se mai modifică semnificativ.

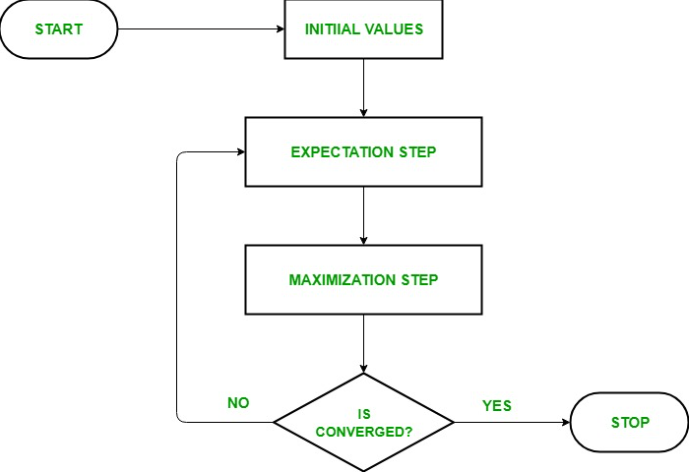


Figura 4 – Schema bloc algoritm

Domenii de aplicabilitate:

* Analiza de date: Completarea datelor lipsă, estimarea distribuției probabile a datelor.
* Învățarea automată: Clustering, clasificare, modelarea amestecurilor de Gaussiene.
* Bioinformatică: Identificarea genelor, analiza secvențelor de ADN.
* Viziunea computerizată: Segmentarea imaginilor, recunoașterea obiectelor.

|  |  |
| --- | --- |
|  | Figura 4.2 – Reprezentare diagrama |

În faza incipientă s-au importat librăriile necsare și s-a cerat setul de date de tip Gaussian cu două componente, după cum este reprezentat în figura 4.2.

Apoi, se efectuează algoritmul EM iterând pentru numărul specificat de epoci, în acest caz, 20. În fiecare epocă, etapa E calculează responsabilitățile (valorile gamma) prin evaluarea densităților de probabilitate gaussiene pentru fiecare componentă și ponderarea lor cu proporțiile corespunzătoare. Etapa M actualizează parametrii prin calcularea mediei ponderate și a deviației standard pentru fiecare componentă, așa cum este reprezentat în figura 4.3

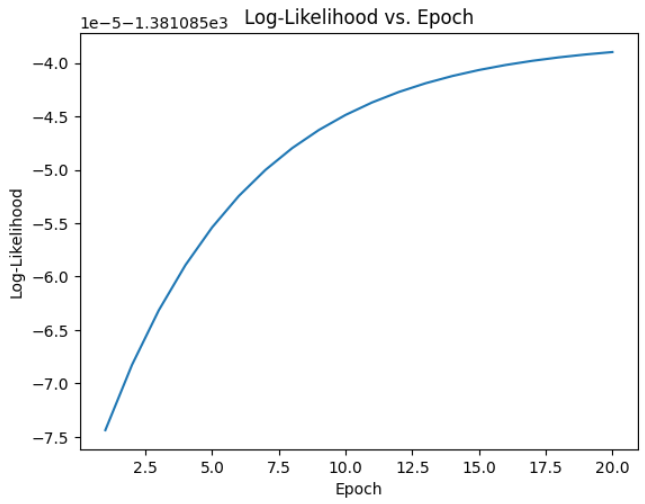


Figura 4.3 – Reprezentare diagramă

În continuare, se sortează vectorul X, iar densitatea estimată finală este calculată prin combinarea funcțiilor de densitate de probabilitate gaussiană pentru fiecare componentă a amestecului, ponderate cu probabilitățile corespunzătoare, fiind reprezentate în figura 4.4.

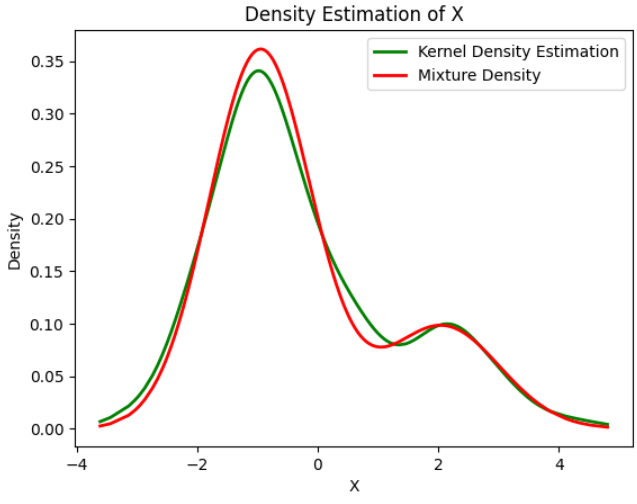


Figura 4.4 – Estimare densitate X

**Concluzie:**

În această lucrare de laborator am abordat clusterizarea datelor prin patru metode distincte: k-Means pentru clustering crisp, algoritmul ierarhic aglomerativ pentru clustering ierarhic, DBSCAN pentru clustering bazat pe densitate și algoritmul Expectation Maximization (EM) pentru clustering probabilistic.

Am folosit algoritmul k-Means pentru a clusteriza datele despre locuințele din California, evidențiind astfel grupurile de case în funcție de locație și prețuri. Determinarea optimă a numărului de clustere (k) a fost o provocare, dar am obținut rezultate satisfăcătoare pentru k=5.

În ceea ce privește algoritmii ierarhici aglomerativi, am implementat o abordare de jos în sus pe datele despre împrumuturi, eliminând valorile aberante și analizând dendrogramele pentru diferite metode de clusterizare.

DBSCAN a fost aplicat pe datele despre clienții unui mall pentru a segmenta clienții în funcție de vârstă, venit și scorul de cheltuieli, identificând eficient 5 clustere și o valoare aberantă.

În final, am folosit algoritmul Expectation Maximization (EM) pe date generate cu două componente de distribuție gaussiană, estimând parametrii modelului și calculând densitatea estimată finală.

În concluzie, clusterizarea este crucială în analiza datelor complexe, dezvăluind pattern-uri și relații. Alegerea corespunzătoare a algoritmilor și parametrilor este esențială pentru obținerea rezultatelor precise în contextul specific al setului de date.