# Dokumentácia ku Klastrovaniu

## Jakub Fridrich

**Zadanie**: Z2b, klastrovanie

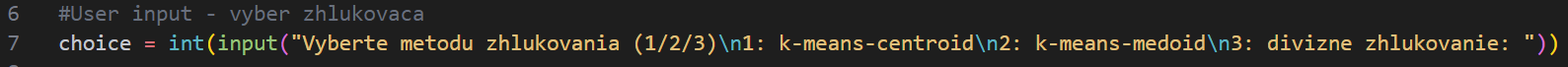
**Použité IDE**: VS Code

**Použité knižnice**: Tkinter, Numpy, Random, Time

**Súbor s kódom**: Z2b.py

## **Vstupné údaje**

Po spustení programu je používateľom zadaná verzia zhlukovača.



## **Reprezentácia údajov**

**Body (Points)**:

**Rozsah súradníc**: Súradnice bodov sa pohybujú v rozsahu od -5000 do 5000.

**Pole array\_points** obsahuje všetky body, ktoré sú generované v rámci zadaného rozsahu súradníc. Každý bod je reprezentovaný dvojicou súradníc (x, y).

**Typ dát**: array\_points je zoznam súradníc, kde každý bod je dvojčlenný zoznam, napríklad [(x1, y1), (x2, y2), ...].

**Centrá klastrov (Centers)**:

**Centroidy a Medoidy**: V závislosti od výberu metódy sa klastre reprezentujú pomocou centroidov alebo medoidov. **Centroid** je priemer všetkých bodov v klastri a nemusí byť jedným z bodov. **Medoid** je skutočný bod v klastri, ktorý minimalizuje priemernú vzdialenosť k ostatným bodom v klastri.

**Inicializácia centier**: Centrá sa inicializujú pomocou **k-means++ metódy**, aby boli centrá rozmiestnené **efektívnejšie**. Táto inicializácia je uložená buď v zozname **centroids\_id** pre centroidy alebo **medoids\_id** pre medoidy.

**Typ dát**: Zoznam dvojíc súradníc podobne ako pri bodoch, ale obsahuje len toľko centier, koľko je požadovaných klastrov (napríklad [(c1\_x, c1\_y), (c2\_x, c2\_y), ...]).

**Klastre (Clusters)**:

Klastre sú reprezentované ako **zoznam zoznamov**. Každý klaster je zoznam bodov, ktoré k nemu patria.

**Pri k-means**: Pre každý klaster sa body priraďujú na základe **najbližšieho centra** (centroidu alebo medoidu). Klastre sa každou iteráciou aktualizujú, kým sa centrá stabilizujú.

**Pri divíznom zhlukovaní**: Klastre sú inicializované s jedným veľkým klastrom ktorý obsahuje všetky body, tento klaster sa následne delí na dva menšie klastre. Proces pokračuje, kým všetky klastre vyhovujú podmienke.

**Typ dát**: Klastre sú reprezentované ako zoznam zoznamov, napríklad [[klaster\_1\_body], [klaster\_2\_body], ...]. Každý vnútorný zoznam obsahuje body patriace do konkrétneho klastra.

**Matica vzdialeností (Distances)**:

Matica vzdialeností vypočítaná vo funkcii **dist\_calc** ukladá vzdialenosti každého bodu k všetkým centrám.

**Typ dát**: Matica je **numpy 2D pole** s rozmermi (num\_points, num\_clus), kde každý riadok predstavuje bod a každý stĺpec vzdialenosť tohto bodu k určitému centru.

**Vizualizácia pomocou Tkinter**:

Program vykresľuje body a centrá klastrov na plátno pomocou knižnice **Tkinter.** Každý bod sa zobrazuje ako malý farebný kruh a centrum klastra je zobrazený ako väčší bezfarebný kruh.

## **Klastrovanie K-means algoritmom**

K-means je algoritmus zhlukovania, ktorý pracuje na princípe priraďovania bodov k *k* rôznym zhlukom (klastrom) podľa ich podobnosti.

**Inicializácia centier:** Náhodne vyberieme *k* bodov v priestore ako počiatočné centrá klastrov (centroidy alebo medoidy). Na efektívnejšie rozmiestnenie centier bude však použité **K-means++**, ktoré je vysvetlené nižšie.

**Priraďovanie bodov:** Každý bod v priestore priradíme k najbližšiemu centru (podľa Euklidovskej vzdialenosti), čím vytvoríme *k* klastrov.

**Aktualizácia centrov:** Pre každý klaster vypočítame nový centroid alebo medoid. Toto nové centrum sa stane novým stredom klastra.

**Opakovanie priraďovania bodov a aktualizácie centrov:** Kroky priraďovania a aktualizácie opakujeme, kým sa centroidy alebo medoidz prestanú výrazne meniť alebo kým algoritmus dosiahne preddefinovanú podmienku.

## **Generovanie bodov**

Po úspešnom výbere zhlukovača sa spustí funkcia **init\_20()**, ktorá vygeneruje 20 unikátnych bodov do 2D priestoru. Tento priestor má rozmery X a Y, v intervaloch od -5000 do +5000.

Následne po vygenerovaní týchto 20 bodov sa spustí funkcia **generate\_more(count)**, ktorá vygeneruje ďalších *count-*bodov, v tomto prípade 40 000. Tieto body sú generované nasledovným spôsobom:

1. Náhodne sa vyberie jeden zo **všetkých doteraz vytvorených** bodov v 2D priestore. (nie len z prvých 20)
2. Vygeneruje sa náhodné číslo **X\_offset** v intervale od -100 do +100
3. Vygeneruje sa náhodné číslo **Y\_offset** v intervale od -100 do +100
4. Pripočíta sa X\_offset a Y\_offset k súradniciam bodu a ak súradnice presahujú 2D priestor, generujú sa nové offsety až kým nebudú korektné.
5. Nový bod sa pridá do 2D priestoru, ktorý bude mať súradnice ako náhodne vybraný bod v kroku 1, pričom tieto súradnice budú posunuté o X\_offset a Y\_offset

Obrázok, na ktorom je text, snímka obrazovky, softvér, multimediálny softvér

Automaticky generovaný popis

## **Inicializácia centier**

Program využíva inicializáciu centier metódou **k-means++.** Táto metóda bola použitá pre efektívne rozmiestnenie centier, čo aj zvyšuje úspešnosť programu.

V prípade, že bolo zvolené **k-means centroid** zhlukovanie, program vygeneruje **náhodné súradnice** v rozhraní súradníc.

Ak však bolo zvolené **k-means medoid** zhlukovanie, program vyberie **náhodný bod** z už existujúcich bodov.

Každý ďalší centroid alebo medoid je vybraný na základe **pravdepodobnosti úmernej vzdialenosti** od najbližšieho už existujúceho centra.

Pre každý bod sa spočíta jeho najmenšia vzdialenosť ku každému už zvolenému centru a na základe týchto vzdialeností sa vytvorí rozdelenie pravdepodobností.

Náhodne sa vyberie nové centrum, kde pravdepodobnosť výberu každého bodu závisí od jeho vzdialenosti (body, ktoré sú ďalej od existujúcich centier, majú väčšiu šancu stať sa novými centrami)

Celý proces zabezpečuje funkcia **kmeans\_pp (arr, k, choice)**

Obrázok, na ktorom je text, snímka obrazovky, softvér

Automaticky generovaný popis

## **Priraďovanie bodov**

Ako prvé sa vypočítajú vzdialenosti bodov a centier pomocou Euklidovej vzdialenosti a rozšírenie o novú os (pre umožnenie broadcastu s **centers\_array**), a to vo funkcii **dist\_calc(array\_centers)**.

Obrázok, na ktorom je text, snímka obrazovky, softvér, písmo

Automaticky generovaný popis

Následne sa pre každý bod nájde index najbližšieho centra vďaka funkcii **clustering(dists)** a priradia sa k príslušnému klasteru pomocou **for-loopu** vo funkcii **kcent\_clustering()** alebo **kmed\_clustering().**

*Ukážka loopu vo funkcii kcent\_clustering():*

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis

*Ukážka loopu vo funkcii kmed\_clustering():*

Obrázok, na ktorom je text, snímka obrazovky, písmo, rad

Automaticky generovaný popis

## **Aktualizácia centier**

Aktualizácia centier vo vašom kóde závisí od toho, či ide o **k-means s centroidmi** alebo **k-means s medoidmi**,

Pri použití **centroidov** ako centier sa nové centrá prepočítavajú na základe priemeru súradníc bodov v každom klastri.

Obrázok, na ktorom je text, písmo, snímka obrazovky

Automaticky generovaný popis

Pri použití **medoidov** ako centier sa nové centrá vyberajú zo skutočných bodov v klastri, aby sa minimalizovala celková vzdialenosť medzi bodmi v rámci klastra

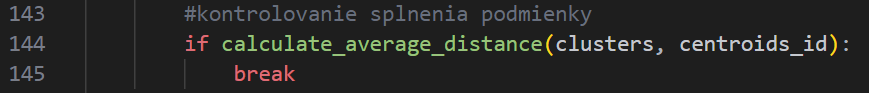
Obrázok, na ktorom je text, písmo, snímka obrazovky

Automaticky generovaný popis

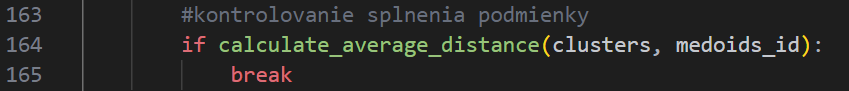
## **Kontrolovanie splnenia podmienky**

Podmienka úspešného ukončenia programu sa v programe kontroluje pomocou funkcie **calculate\_average\_distance**. Táto funkcia vypočíta **priemernú vzdialenosť bodov od centra** v každom klastri. Ak sú všetky priemerné vzdialenosti v rámci klastrov **menšie alebo rovné stanovenému prahu (500)**, algoritmus sa považuje za **úspešne ukončený**, pretože sa dosiahla dostatočná zhlukovacia presnosť. Ak nie je úspešne dosiahnutý prah do limitu iterácii, program sa považuje za neúspešný.

*Volanie vo funkcii kcent\_clustering():*



*Volanie vo funkcii kmed\_clustering():*

**

Obrázok, na ktorom je text, multimediálny softvér, softvér, snímka obrazovky

Automaticky generovaný popis

## **Klastrovanie Divíznym zhlukovaním**

Divízne zhlukovanie na princípe **postupného delenia klastrov**, pričom začíname s jedným veľkým klastrom obsahujúcim všetky body a postupne ho rozdeľujeme na menšie klastre, kým každý klaster nedosiahne požadovanú presnosť, to je v každom klastri priemernú vzdialenosť bodov od stredu viac ako 500.

**Inicializácia:** Vytvorí sa počiatočný klaster obsahujúci všetky body.

**Výpočet priemernej vzdialenosti**: Pre každý klaster sa vypočíta centroid (stred klastra) a priemerná vzdialenosť všetkých bodov v tomto klastri od jeho centroidu.

**Rozdelenie klastrov**: Ak je priemerná vzdialenosť bodov od centroidu väčšia než stanovený prah, klaster sa rozdelí na dva menšie klastre.

**Opakovanie**: Tento proces sa opakuje pre každý nový klaster, kým všetky klastre nesplnia podmienku maximálnej priemernej vzdialenosti.

**Ukončenie**: Ak sú všetky klastre dostatočne kompaktné, algoritmus sa zastaví.

## **Inicializácia a Výpočet priemernej vzdialenosti**

Obrázok, na ktorom je text, snímka obrazovky, softvér

Automaticky generovaný popis

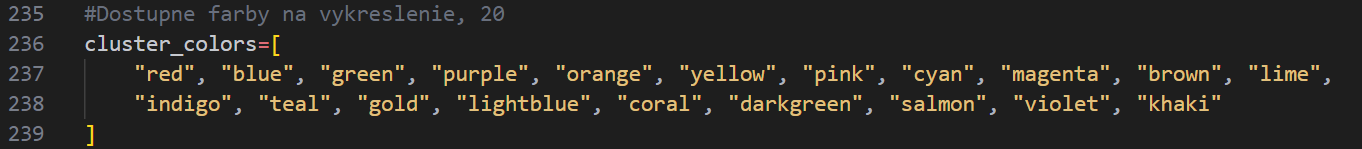
## **Rozdelenie klastrov**

Obrázok, na ktorom je text, snímka obrazovky, softvér

Automaticky generovaný popis

## **Vizualizácia**

Vizualizácia je realizovaná funkciou **draw\_clusters(),** ktorá je volaná v každom type zhlukovača. Táto funkcia vykresli všetky body a finálne klastre, pričom každý bod je farebne odlíšený farbami zo zoznamu **cluster\_color.** Každý bod je transformovaný pomocou škálovacieho faktora (**scaling\_down=20**) a posunutý do stredu plátna tak, aby boli všetky body viditeľné.



Obrázok, na ktorom je text, snímka obrazovky

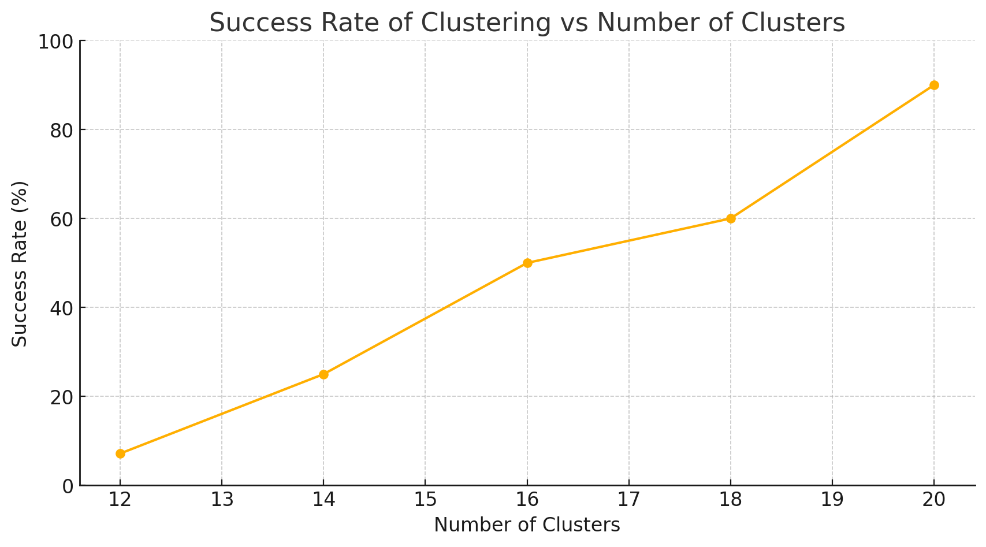
Automaticky generovaný popis

## **Porovnanie výsledkov**

Pre demonštráciu kódu som sa rozhodol otestovať každý typ zhlukovača a porovnať získané výsledky. Testovanie zhlukovačov k-means prebiehalo pri nasledovných podmienkach:

Pri K-means, kde stred je centroid bolo generovaných 40 000 bodov, limita iterácií bola 50 a celkovo bolo urobených 20 pokusov.

Graf úspešnosti klastrovania (centroid) vzhľadom ku počtu klasterov vyzerá nasledovne:

****

S priemernými časmi pokusov pre jednotlivé počty klastrov: 12 klastrov: **15.83 sekúnd**

1. klastrov: **19.24 sekúnd**

16 klastrov: **22.72 sekúnd**

18 klastrov: **29.61 sekúnd**

20 klastrov: **36.38 sekúnd**

Pri K-means, kde stred je medoid bolo taktiež generovaných 40 000 bodov, avšak limita iterácií bola iba 10 a celkovo bolo urobených 20 pokusov.

Graf úspešnosti klastrovania (medoid) vzhľadom ku počtu klasterov vyzerá nasledovne:

Obrázok, na ktorom je text, rad, diagram, vývoj

Automaticky generovaný popis

S priemernými časmi pokusov: 12 klastrov: **139.12 sekúnd**

14 klastrov: **143.4 sekúnd**

16 klastrov: **83.65 sekúnd**

18 klastrov: **42.24 sekúnd**

20 klastrov:**44.36 sekúnd**

V prípade Divízneho zhlukovania sa testoval program 10 krát na 40 000 bodoch. Testovaním sa zistilo, že priemerný čas pracovania je **9.98 sekúnd** a priemerný počet klastrov je **13.1.**

Po porovnaní týchto výsledkov je možné vidieť, že z hľadiska časovej efektívnosti a úspešnosti je metóda Divízneho zhlukovania pre podmienky Zadania 2 najvhodnejšia.

Čo sa týka úspešnosti, po divíznom zhlukovaní nasleduje k-means medoid, ktorý je od 16 klasterov úspešný viac ako 75% pokusov. Avšak k-means centroid môžeme považovať za síce rýchlejšiu, ale menej úspešnú metódu