Zadanie č. 2: klasifikácia

V mojom zadaní som použil K-NN na definovanie farby bodov podľa K najbližších susedov (1, 3, 7, 15)

Použitý algoritmus:

Každý bod je podľa súradníc pridelený do bloku v mriežke. Pri klasifikácii pozrie na svoj blok a okolité bloky (vpravo, vľavo, hore, dole a diagonálne) týmto prehľadáva 3x3 blokov kde hľadá susedov

Ak v 3x3 mriežke nenájde dostačujúci počet susedov teda K ide vyhľadávať do 5x5 blokov okolo seba

Ak ani v 5x5 nenájde dostatočný počet bodov prejde na algoritmus brute-force v ktorom porovnáva vzdialenosť s každým už klasifikovaným bodom

Ukladanie údajov:

Generated\_points= [] : pole obsahujúce prvotnú generáciu prvkov a ich X,Y

Inital\_classes= [] : pole obsahujúce prvotné zaradenie tried pri generácii

Initial\_starting\_points={} : knižnica obsahujúca prvých 5 bodov pre každú triedu zo zadania

Casses=[]: pole tried

Class\_as\_color: knižnica ktorá prideľuje každej triede farbu

Grid\_points= defaultdict(list): knižnica v ktorej počas behu programu sú pridelené body pre dané bloky v mriežke

Dist\_cls= []: pole tuplov kde sa ukladá vzdialenosť od klasifikovaného bodu a trieda bodu od ktorého klasifikujem

Str\_p={} : knižnica do ktorej je kopírovaná knižnica Initial\_starting\_points (aby sme o ňu neprišli) a sú do nej pridávané všetky novo klasifikované body kvôli vypíšu a brute-force

Most\_common\_class : drží v sebe najbežnejšiu triedu z K susedov

Stt\_p= {} : knižnica obsahujúca všetky novo už klasifikovane body na vizualizáciu

Zistenie a práca so vzdialenosťami:

Na zistenie vzdialenosti používam Euklidov algoritmus:

Obrázok, na ktorom je text, písmo, snímka obrazovky

Automaticky generovaný popis

Na usporiadanie poľa vzdialeností a zistenie najbežnejšej triedy používam sort(lambda) a Counter:

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis

Použité knižnice:

* Import Matplotlib: vykresľovanie výsledkov
* Import Random: vytváranie náhodných čísel a vyberanie náhodných členov
* Import Numpy: knižnica na numerické operácie nad viacrozmernými poľami
* From collection import Counter, defaultdict:
  + Counter: jednoduché počítanie výskytu rôznych prvkov v zozname
  + Defaultdict: automatické priradenie automatickej hodnoty pre nové kľúče v slovníku
* Import time: výpočet času behu programu

Generovanie bodov:

Body generujem sequnečne aby som zaistil že každá farba ma rovnaký počet bodov

V poradí : Red, Green, Blue, Purple

Vykraslenie :

Výsledok vykresľujem v 2x2 oknách z knižnice stt\_p podľa farieb Red, Green, Blue, Pruple (vykresľovanie vedie k prekrytiu inak zakrytých bodov)

Tento typ vykresľovania som zvolil z dôvodu časovej optimalizácie keďže ak som vykresľoval bod akonáhle som ho klasifikoval tak program bežal niekoľko násobne dlhšie

Postup behu programu:

1. Generácia bodov:
   1. Funkcia start() v cykle definuje triedy a vola generate\_points
   2. Funkcia generate\_points() vygeneruje bod podľa triedy a šumu
2. Klasifikácia:
   1. Volá sa funkcia zobraziť()
      1. Obsahuje cyklus pre všetky k členy
   2. V cykle sa vyberajú z poľa generated\_points() členy na klasifykáciu
   3. Volá sa funkcia classiffy() ktorá pre bod vracia triedu
   4. Všetky novo klasifikované body sa uložia do poľa a vykreslia sa

Výsledky:

|  |  |
| --- | --- |
| 1 | 3 |
| 7 | 15 |

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popisObrázok, na ktorom je pestrofarebnosť

Automaticky generovaný popis

|  |  |
| --- | --- |
| 1 | 3 |
| 7 | 15 |

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popisObrázok, na ktorom je pestrofarebnosť, vzor

Automaticky generovaný popis

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis

|  |  |
| --- | --- |
| 1 | 3 |
| 7 | 15 |

Obrázok, na ktorom je pestrofarebnosť, štvorec

Automaticky generovaný popis