Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológii

Zadanie 4a

Umelá Inteligencia

Martin Raffáč

Zadanie

Máme 2D priestor, ktorý má rozmery X a Y, v intervaloch od -5000 do +5000. V tomto priestore sa môžu nachádzať body, pričom každý bod má určenú polohu pomocou súradníc X a Y. Každý bod má unikátne súradnice (t.j. nemalo by byť viacej bodov na presne tom istom mieste). Každý bod patrí do jednej zo 4 tried, pričom tieto triedy sú: red (R), green (G), blue (B) a purple (P). Na začiatku sa v priestore nachádza 5 bodov pre každú triedu (dokopy teda 20 bodov). Súradnice počiatočných bodov sú:

R: [-4500, -4400], [-4100, -3000], [-1800, -2400], [-2500, -3400] a [-2000, -1400]

G: [+4500, -4400], [+4100, -3000], [+1800, -2400], [+2500, -3400] a [+2000, -1400]

B: [-4500, +4400], [-4100, +3000], [-1800, +2400], [-2500, +3400] a [-2000, +1400]

P: [+4500, +4400], [+4100, +3000], [+1800, +2400], [+2500, +3400] a [+2000, +1400]

Vašou úlohou je naprogramovať klasifikátor pre nové body – v podobe funkcie classify(int X, int Y, int k), ktorá klasifikuje nový bod so súradnicami X a Y, pridá tento bod do nášho 2D priestoru a vráti triedu, ktorú pridelila pre tento bod. Na klasifikáciu použite k-NN algoritmus, pričom k môže byť 1, 3, 7 alebo 15.

Na demonštráciu Vášho klasifikátora vytvorte testovacie prostredie, v rámci ktorého budete postupne generovať nové body a klasifikovať ich (volaním funkcie classify). Celkovo vygenerujte 20000 nových bodov (5000 z každej triedy). Súradnice nových bodov generujte náhodne, pričom nový bod by mal mať zakaždým inú triedu (dva body vygenerované po sebe by nemali byť rovnakej triedy):

- R body by mali byť generované s 99% pravdepodobnosťou s X < +500 a Y < +500
- G body by mali byť generované s 99% pravdepodobnosťou s X > -500 a Y < +500
- B body by mali byť generované s 99% pravdepodobnosťou s X < +500 a Y > -500

- P body by mali byť generované s 99% pravdepodobnosťou s X > -500 a Y > -500

  (Zvyšné jedno percento bodov je generované v celom priestore.)

Návratovú hodnotu funkcie classify porovnávajte s triedou vygenerovaného bodu. **Na základe týchto porovnaní vyhodnoťte úspešnosť** Vášho klasifikátora pre daný experiment.

Experiment vykonajte 4-krát, pričom zakaždým Váš klasifikátor použije iný parameter k (pre k = 1, 3, 7 alebo 15) a vygenerované body budú pre každý experiment rovnaké.

Vizualizácia: pre každý z týchto experimentov vykreslite výslednú 2D plochu tak, že vyfarbíte túto plochu celú. Prázdne miesta v 2D ploche vyfarbite podľa Vášho klasifikátora.

Dokumentácia musí obsahovať opis konkrétne použitého algoritmu a reprezentácie údajov. V závere zhodnoťte dosiahnuté výsledky ich porovnaním.

Algoritmus a riešenie

1. Na začiatku sa zadáva počet bodov ktoré má program generovať pre jednotlivé farby to znamená že keď je počiatočné číslo NUM 5000 program vygeneruje 5000 bodov pre červenú zelenú modrú a fialovú to je dokopy 20000 bodov plus počiatočných 20 bodov ktoré sú preddefinované, takže dokopy program vygeneruje 20020 bodov.

2. Keďže pre každé **K** sa majú použiť rovnaké body program ich vygeneruje vopred. Tieto body sa generujú tak aby sa tam nenachádzali žiadne duplicity a aby existovalo 5000 správnych. Následne program vytvorí 5000 bodov v nesprávnom rozpätí aj tieto body sú potrebné keďže program má byť presný len na 99%.

3. Následne sa for cyklus vykoná 4 krát pre k: 1, 3, 7, 15. Pre každé k sa vykoná tento algoritmus: vyberie sa náhodná farba z farieb: red, green, blue, purple (vyberá sa taká ktorá nebola naposledy vybraná). Po pridelení farby sa vyberá pravdepodobnosť, či je tento bod zo správneho rozpätia. Podľa toho či je správny alebo nesprávny sa vyberá bod z daných polí. Pripočíta sa počet výskytov tejto farby a zavolá sa funkcia classify s

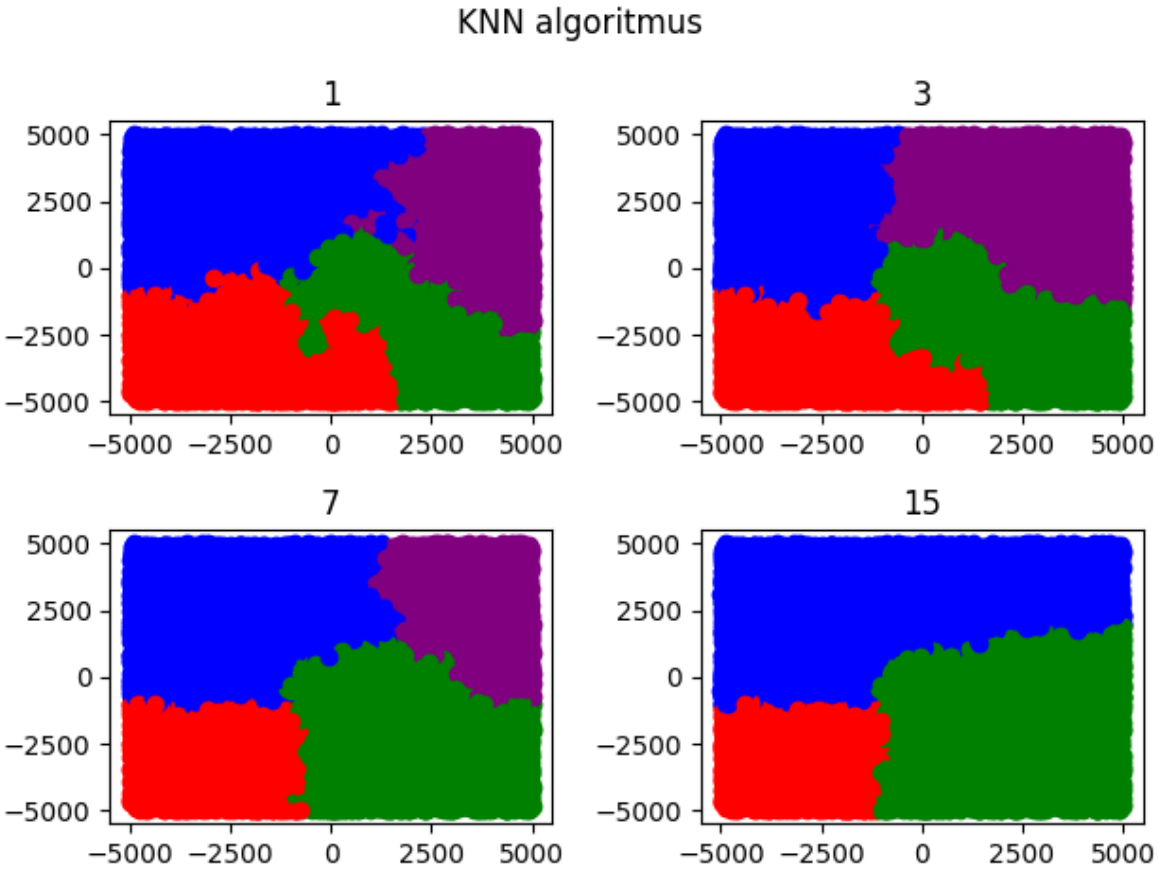
parametrami x, y, k v ktorej sa počíta euklidovská dĺžku od x a y súradnice môjho terajšieho bodu ktorý chcem klasifikovať so všetkými tréningovými bodmi ktoré už boli klasifikované. Následne sa usporiadajú hodnoty od najmenšej vzdialenosti. Program následne vyberie k najbližších a podľa nich sa určí farba tohto bodu. Tento bod sa pridá do tréningových dát. Následne sa porovnáva farba ktorú pridelila funkcia classify s farbou ktorú tento bod mal mať. V prípade, že sa farby rovnajú program nerobí nič, keď sa ale farby nerovnajú pripočítava sa chyba. V momente keď už je počet bodov každej 5000 program končí.

4. Nakoniec sa už len vypíše čas, ako dlho trvalo riešenie a koľko chýb sa. Graf sa vykresľuje pomocou knižnice plt. Program vykreslí 4 grafy pre každé k.

Používateľské rozhranie

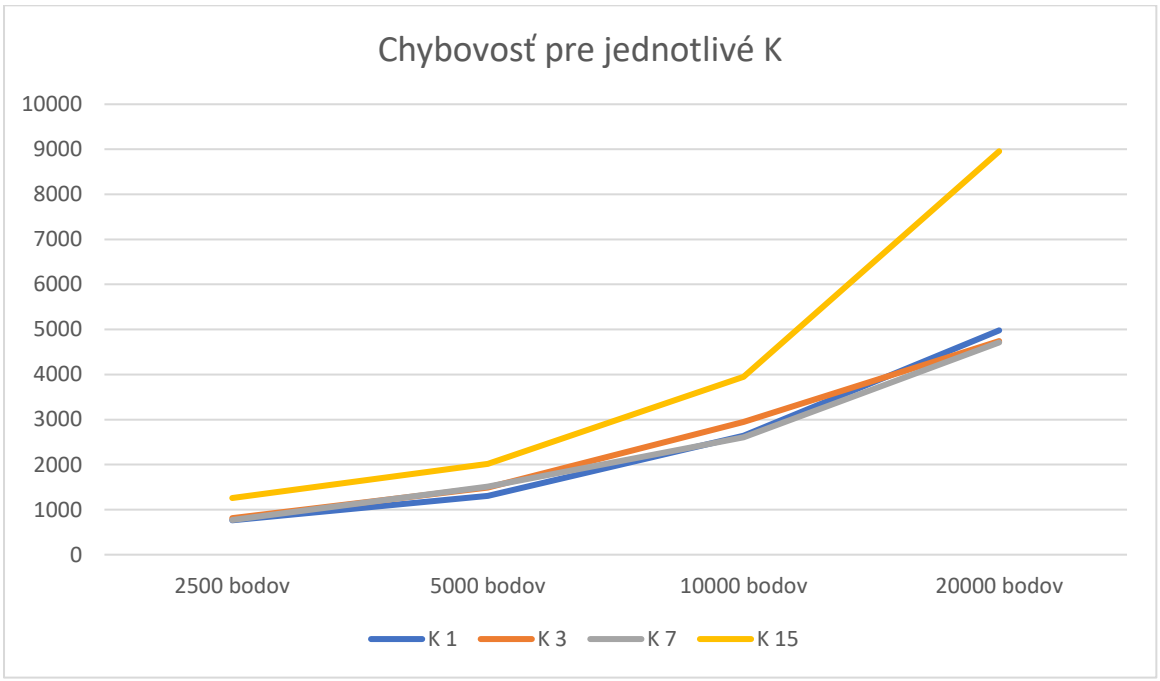
```
Pocet bodov je: 20020
pocet chyb je:  5017
cas trvavia generovania: 160.44453406333923
pocet chyb je:  4634
cas trvavia generovania: 169.99615669250488
pocet chyb je:  4869
cas trvavia generovania: 178.71293926239014
pocet chyb je:  11675
cas trvavia generovania: 170.1274607181549
koniec programu
```

Výsledný graf vyzerá nasledovne:

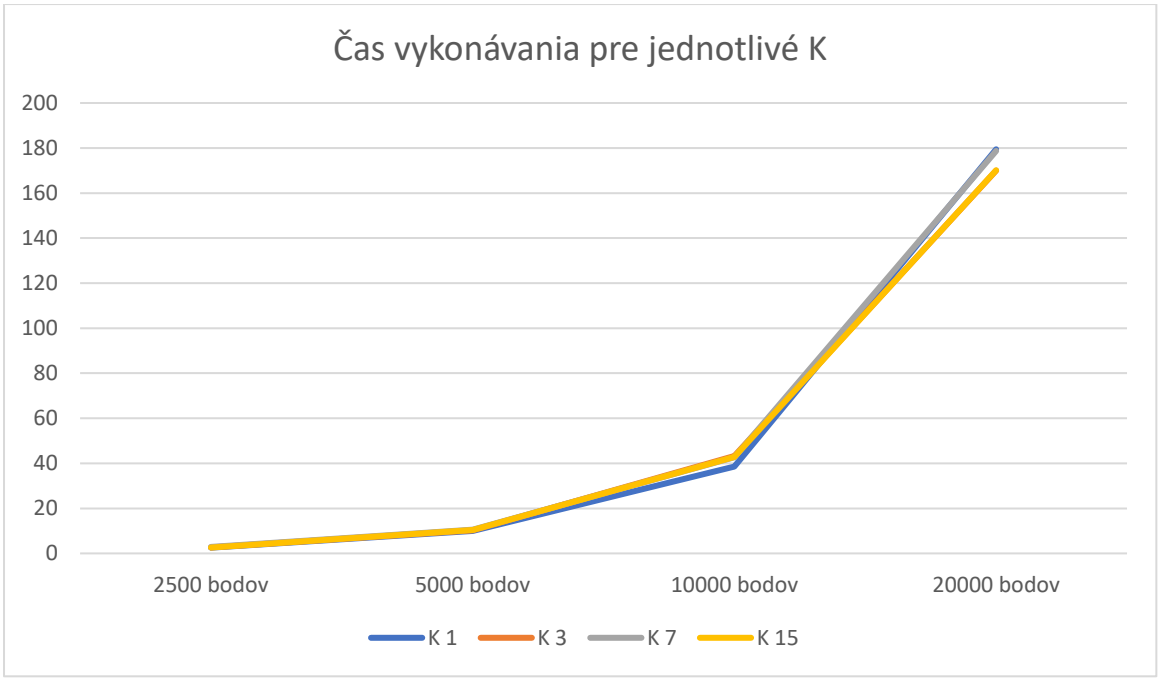


Testovanie

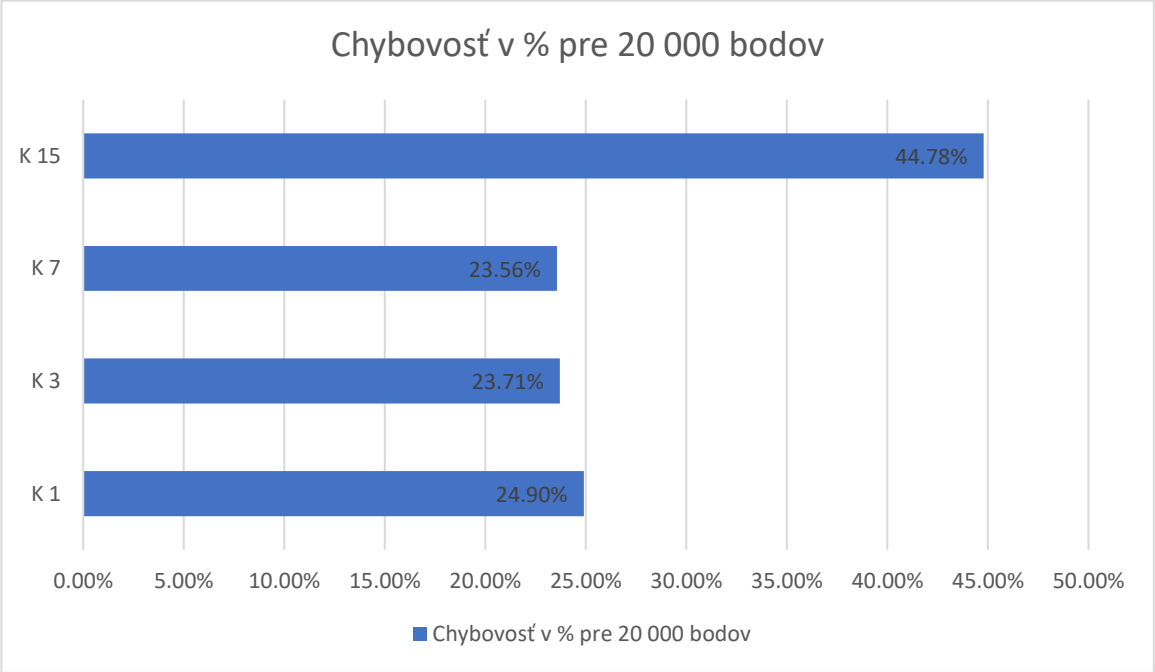
Testy som vykonal 4 krát ako bolo požadované v zadaní pre rôzny počet vygenerovaných bodov.

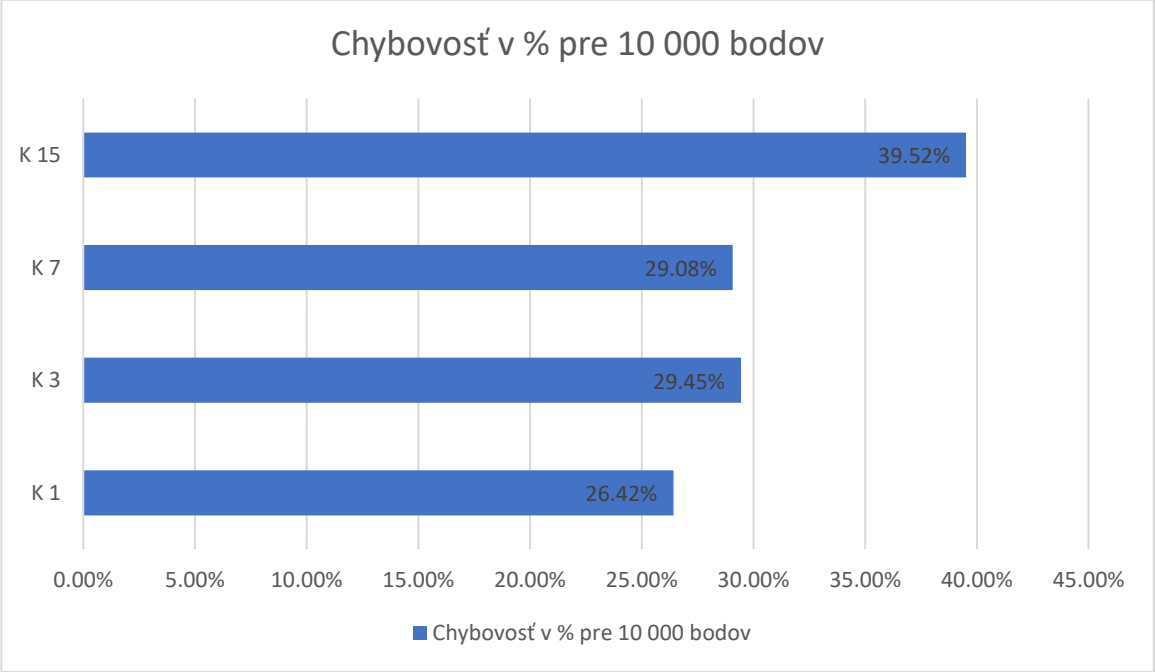


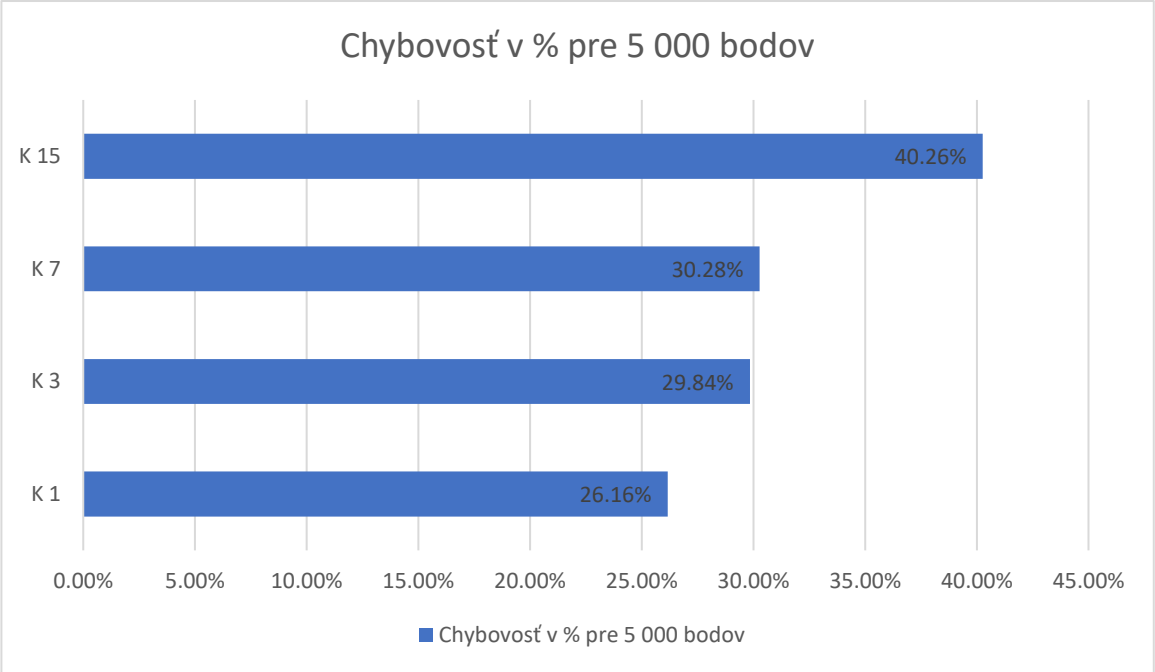
Z grafu jednoznačne vyplýva, že so stúpajúcim počtom bodov stúpa aj chybovosť. Pričom najmenej efektívny je algoritmus pri k = 15

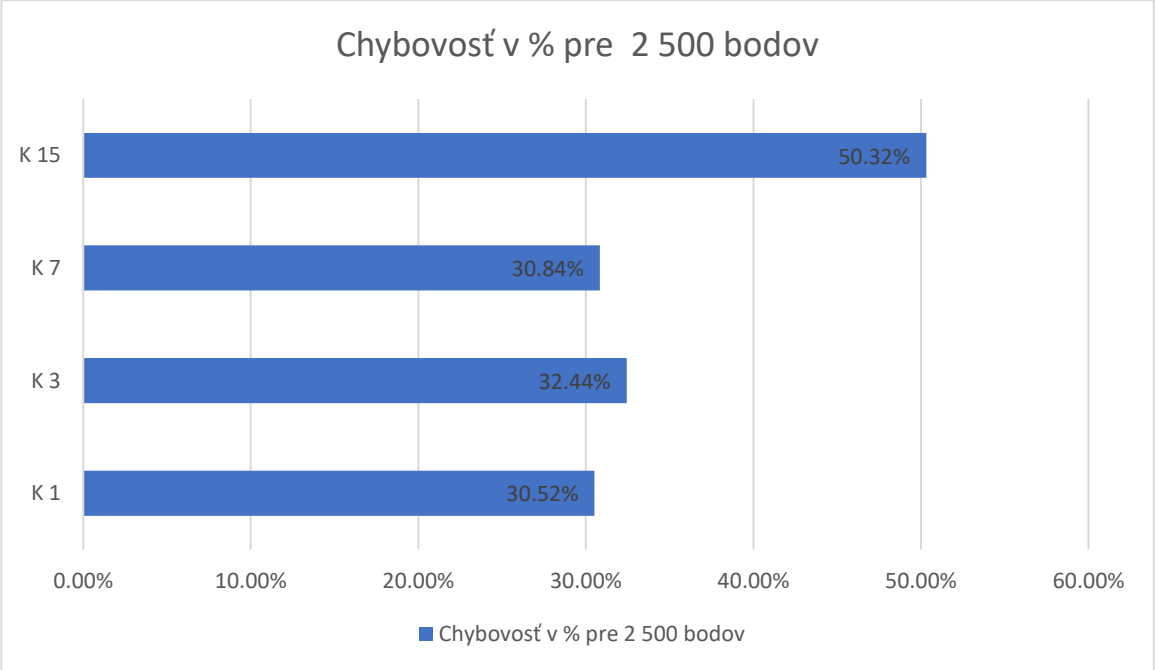


Z tohto grafu vyplýva že čas na vykonanie programu sa zvyšuje so zvyšujúcim sa K. A medzi jednotlivými K nie je veľký rozdiel pri dĺžke trvania algoritmu.

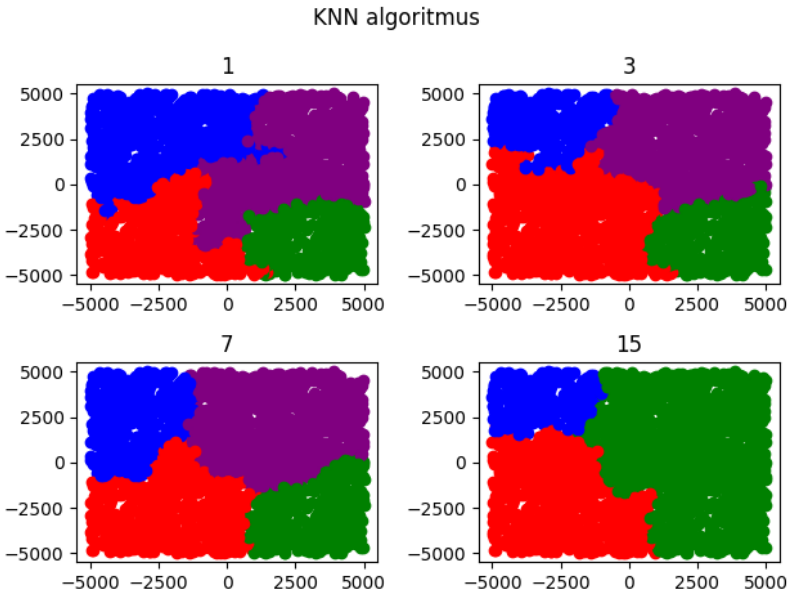




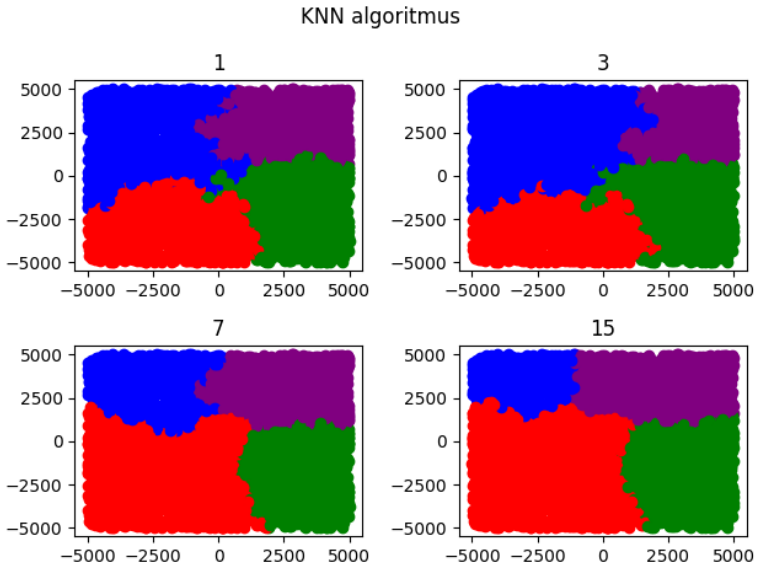

Chybovosť v % pre 5 000 bodov
K 15
K 7
K 3
K 1
40.26%
30.28%
29.84%
26.16%
0.00%
5.00%
10.00%
15.00%
20.00%
25.00%
30.00%
35.00%
40.00%
45.00%
Chybovosť v % pre 5 000 bodov


Chybovosť v % pre 2 500 bodov
K 15
K 7
K 3
K 1
50.32%
30.84%
32.44%
30.52%
0.00%
10.00%
20.00%
30.00%
40.00%
50.00%
60.00%
Chybovosť v % pre 2 500 bodov

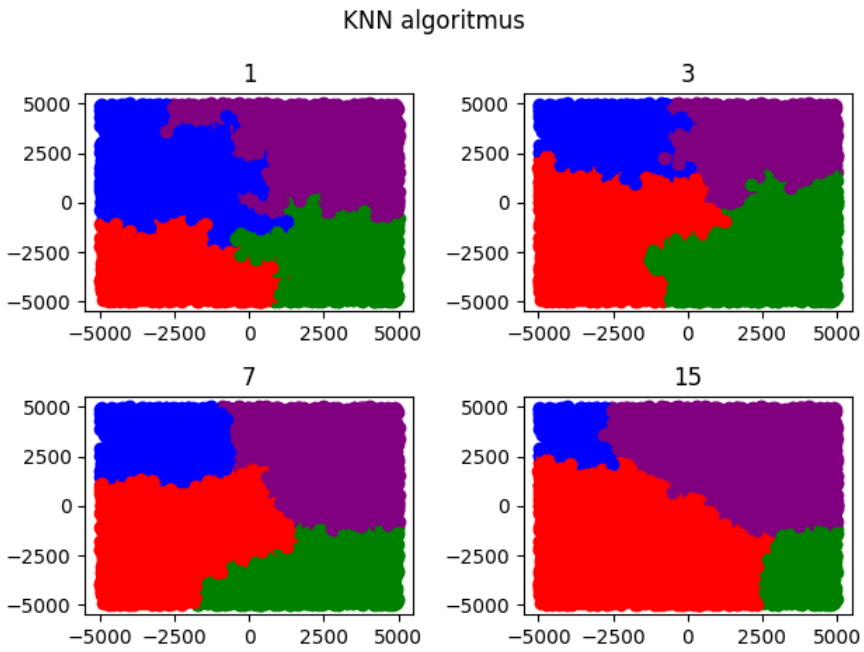
2 500 bodov



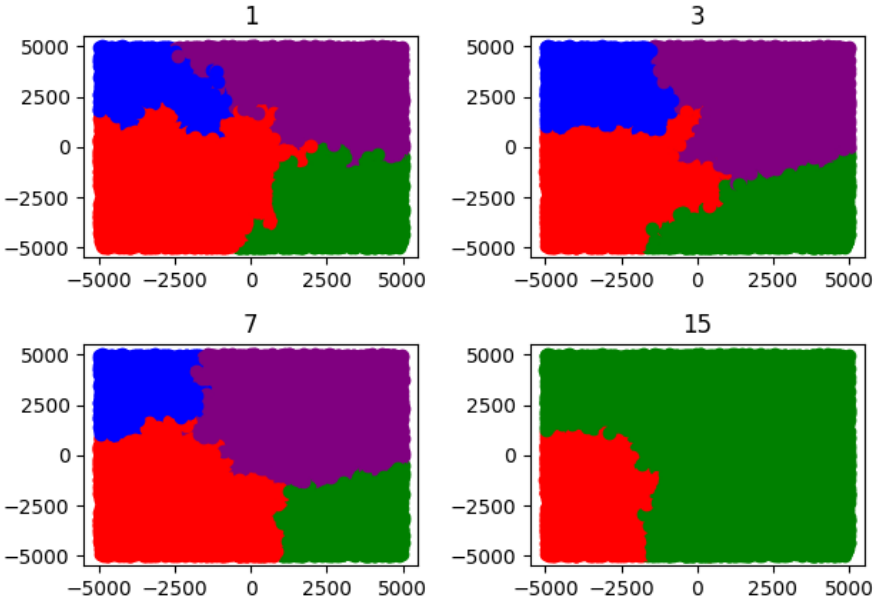
5 000 bodov



10 000 bodov



20 000 bodov



Hodnotenie

Mnou navrhnutý program vie vyriešiť KNN algoritmus pre dané K podľa testovania pre 20 000 bodov maximálne do zhruba 200 sekúnd, čo by som po porovnaní so spolužiakmi nazval ako správne riešenie. Z experimentov ktoré som vykonal jasne vyplýva, algoritmus je najúspešnejší keď K=7 a naopak najmenej úspešný pri K=15. Program som implementoval v pythone