FIIT STU

Umelá inteligencia

Zadanie 4a - Klasifikácia

Cvičenie: Utorok 14:00-15:40, Kapustík

Adam Miškove

Obsah

1. Opis môjho riešenia problému

A) Začiatok programu a generácia bodov:

Po spustení programu sa vypýta jednoduchý vstup - počet vygenerovaných bodov(z každej triedy). Po úspešnom zadaní vstupu sa zapne hlavná funkcia programu, ktorá všetko riadi, **mainLoop()**.

Na jej samom začiatku sa vygeneruje počiatočných 20 bodov(5 pre každú triedu), ako je spomenuté v zadaní. O toto sa postará funkcia **initializePoints().** Tá opakovane volá funkciu **fiveStartingPoints()** s argumentmi reprezentujúcimi znamienko pred x,y súradnicami. Táto funkcia nasledovne vytvorí dané body a pridá ich do listu **points**.

Po úspešnej inicializácii začiatočných bodov sa program presunie na funkciu **generateNewPoints()**, ktorá kompletne ovláda generáciu nových bodov a ich priraďovanie do listu s bodmi points. Takisto pri každom vytvorení bodov sa tieto body pridávajú do rozdelenia nášho listu na podlisty **splinters**, pričom lokácia bodu v týchto podlistoch je určená jeho súradnicami.

Tieto podlisty istým spôsobom rozdeľujú našu pomyselnú plochu na 25 podplôch(teda na plochy s rozmermi po 2000x2000), a pri vyberaní najbližších bodov pri KNN algoritme sa vyberá z okolitých/priľahlých podplôch, relatívne k súradniciam porovnávaného bodu. Body sú generované s 99% šancou na súradnice v intervaloch podľa zadania, a 1% šancou na súradnice v intervale <-5000,5000>. Body sa do podplôch pridávajú funkciou **createSplinter(),** ktorá nájde cieľovú podplochu pomocou funkcie **findSplinter().**

```
def createSplinter(point, splinters):
    x,y = findSplinter(point[0],point[1])
    splinters[x][y].append(point)

def findSplinter(x,y):
    x+=5000
    y+=5000
    xn = int(x/2000)
    if(xn==5):
        xn=4
    yn = int(y / 2000)
    if(yn==5):
        yn=4
    return xn,yn
```

V rámci môjho programu sú body reprezentované ako listy s obsahom = [xSúradnica, ySúradnica, vygenerovaná trieda, trieda klasifikovania s k=1, trieda klasifikovania s k=3, trieda klasifikovania s k=7, trieda klasifikovania s k=15]. Obvykle sa vytvárajú funkciou **createPoint()**.

```
def createPoint(x,y,clxss, clxss1, clxss3, clxss7, clxss15):
    point = [x,y, clxss, clxss1, clxss3, clxss7, clxss15]
    return point
```

Spomenutá funkcia **generateNewPoints()** zavolá n-krát funkciu **generateFourPoints()**, ktorá zakaždým podľa spomínaných pravidiel vygeneruje 4 body. Tieto body sú samozrejme generované v poradí red->green->blue->purple.

Teda pre 20 000 vygenerovaných bodov zadávame n=5000. Pri generovaní jednotlivých bodov je samozrejme potrebné overiť ich unikátnosť, teda volá sa funkcia **checkPoint()**, ktorá vracia True, ak je bod unikátny, a False, ak nie je.

```
def checkPoint(points, x, y, splinters):
    a,b = findSplinter(x,y)
    for i in range(0,len(splinters[a][b])):
        if(splinters[a][b][i][0] == x and splinters[a][b][i][1] == y):
            return False
    return True
```

B) Klasifikácia bodov:

Po úspešnom vytvorení bodu sa prechádza na časť klasifikácie. Tá začína vo funkcii **classify()**, ktorá nájde okolité(priľahlé) podplochy k nášmu klasifikovanému bodu. Nasledovne vytvorí list **distances**, do ktorého sa budú pridávať informácie relevantné ku klasifikácii, teda vzdialenosť, a klasifikácie každého druhu(základná, k=1,k=3,k=7,k=15).

Klasifikácie pre všetky štyri hodnoty **k** sa konajú v jednom behu rodičovskej funkcie, keďže je potrebné ich vykonať na rovnako generovanej množine bodov. Spomenutý list **distances,** obsahujúci vzdialenosť+všetky triedy daného bodu(pred ich klasifikáciou sú 4 klasifikačné triedy nastavené na základnú hodnotu, ktorá bola určená pri vytvorení bodu), bude usporiadaný vzostupne(od najmenšieho po najväčšie), podľa svojej informácie vzdialenosti medi bodmi.

Keď máme list so vzdialenosťami usporiadaný, môžeme volať druhú klasifikačnú funkiu, **classifyFinal().** Táto funkcia spočíta počet výskytov každej triedy bodov v najbližších k-tich.

```
for i in range(0,len(distances)):
    if (distances[i][n] == "red"):
        r += 1
    if (distances[i][n] == "green"):
        g += 1
    if (distances[i][n] == "blue"):
        b += 1
    if (distances[i][n] == "purple"):
        p += 1
```

Po vykonaní tejto úlohy je potrebné vybrať všetky triedy, ktorých počet výskytov je maximum spočítaných výskytov.

Toto znamená, že ak máme príklad s:
k=7,
r=1,
g=0,
b=3,
p=3,
Vyberú sa triedy(do listu **maxcounts**), ktorých je maximálny nájdený počet(3 pre triedy blue a purple).

```
counts = [r, g, b, p]
maximum = max(counts)
maxcounts = []
for i in range(0,len(counts)):
    if(counts[i]==maximum):
        if (i == 0):
            maxcounts.append("red")
        if (i == 1):
            maxcounts.append("green")
        if (i == 2):
            maxcounts.append("blue")
        if (i == 3):
            maxcounts.append("purple")
```

Posledný krok klasifikácie je prejdenie cez list so vzdialenosťami, a nájdenie prvého výskytu farby/triedy v liste **maxcounts**, teda snažíme sa nájsť najbližší výskyt triedy, ktorej bolo maximum. Pri nájdení prvého vyhovujúceho bodu sme našli korektnú triedu pre klasifikáciu nášho bodu, a môžeme ju vrátiť.

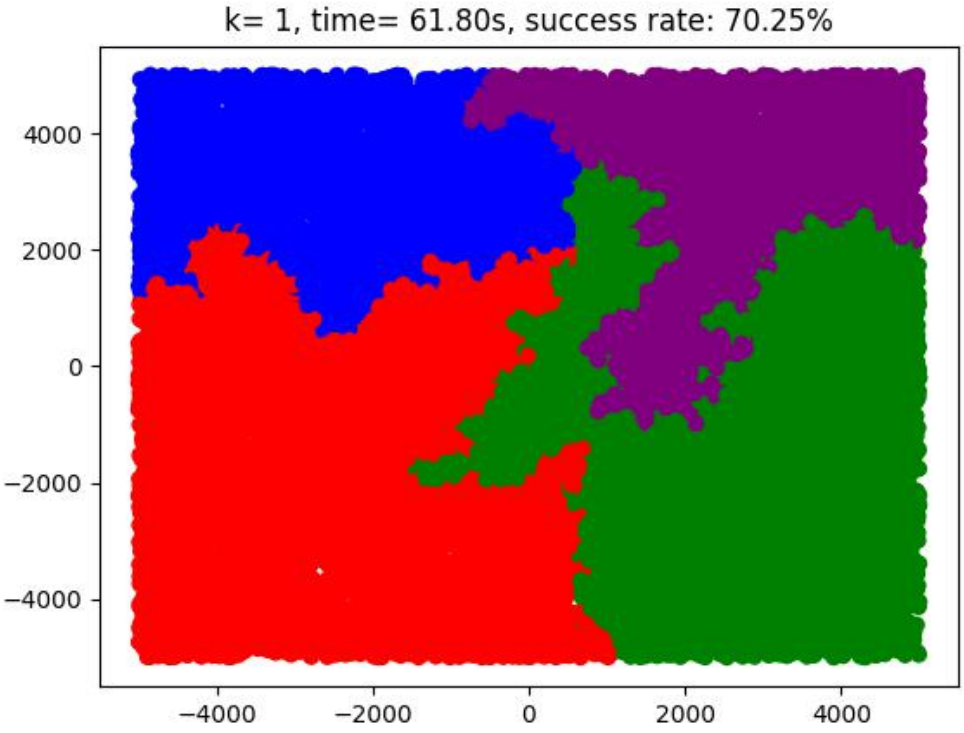
```
for i in range(0,len(distances)):
    for j in range(0,len(maxcounts)):
        if(distances[i][n] == maxcounts[j]):
            found = 1
            classifiedAs = maxcounts[j]
            break
    if(found == 1):
        break

return classifiedAs
```

Nájdené výsledky sa uložia do listu **classifications**, ktorý obsahuje dáta v podobnom formáte ako list **points,** lenže neobsahuje informácie o prvých 20tich bodov, ktoré nás nezaujímajú pri finálnom vizualizovaní výsledkov pomocou scatterplotu.

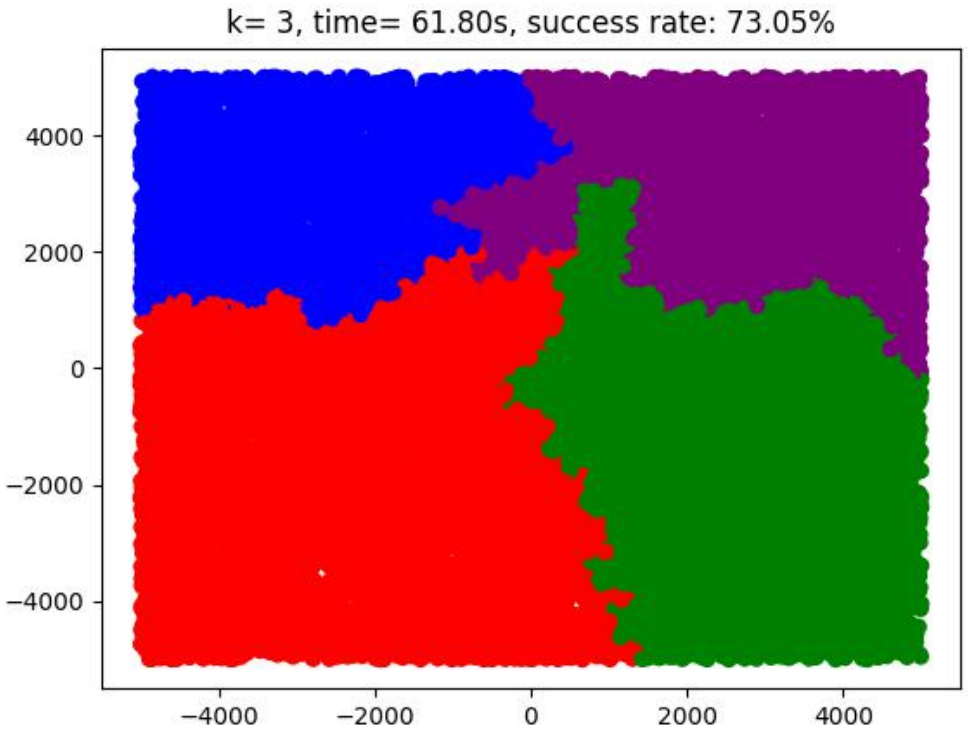
2) Vizualizácia výsledkov

Nižšie uvádzam vizualizáciu výsledných dát v podobe scatterplotu z pythonovskej knižnice **matplotlib.pyplot**.
Prirodzene sú tieto výsledky pre 20 000 vygenerovaných bodov.

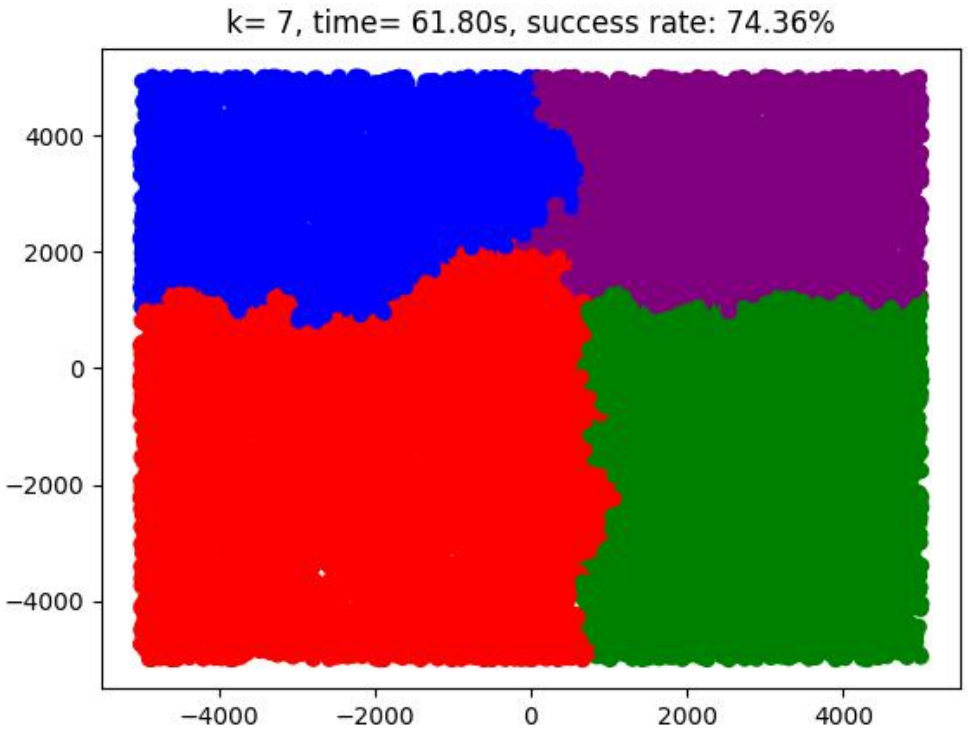
Výsledok pre k=1.
Pre takto nízku hodnotu k je vyššia šanca, že sa bod vygeneruje hneď vedľa bodu s inou triedou, teda automaticky sa klasifikácia triedy bude líšiť.



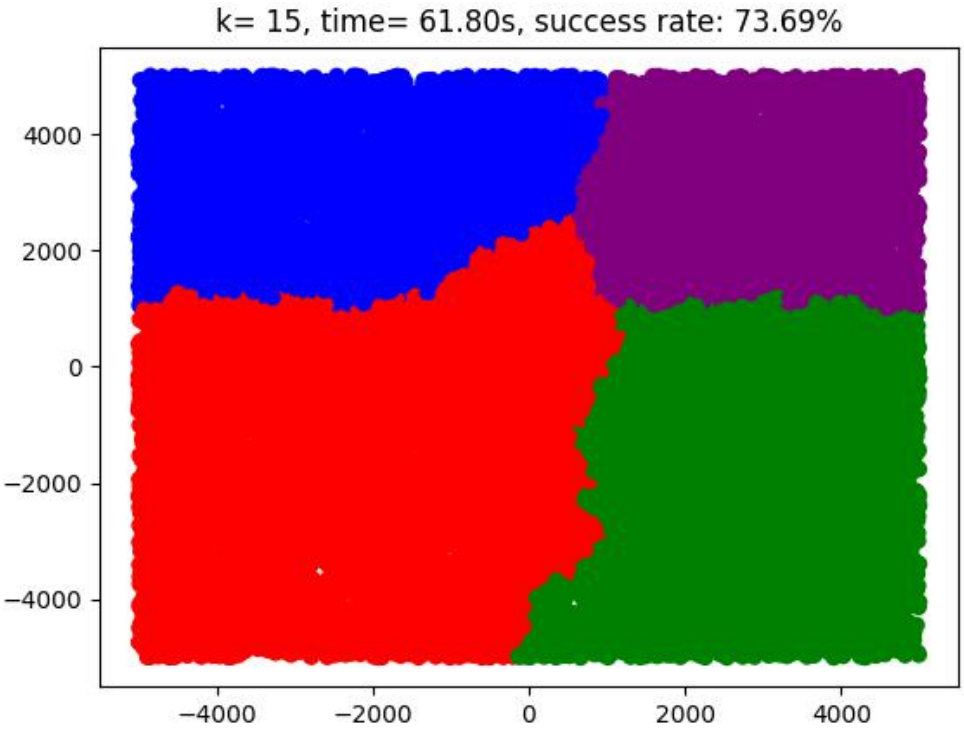
Výsledok pre k=3.
Zvýšením hodnoty k sa znižuje šanca nešťastného vygenerovania bodu s najbližším iným bodom inej triedy. Toto znamená, že šanca správnej klasifikácie sa zdvihne.



Výsledok pre k=7.
Opakuje sa trend z minulého pozorovania, teda miernym zvýšením hodnoty k sa takisto mierne zvýši hodnota úspešnej klasifikácie.



Výsledok pre k=15.
Pri takto prudkom zvýšení hodnoty k, teda zo 7 na 15, je možno pozorovať mierny pokles úspešnosti klasifikácie, čo by sa dalo vysvetliť tým, že stúpajúcou hodnotou k takisto stúpa šanca pre nesprávnu klasifikáciu v prípade, že je daný bod vygenerovaný na mieste, kde je veľa zmiešaných tried, teda je vyššia šanca, že sa vyberie trieda, do ktorej iniciálne nepatrí.



3) Záver

Okrem porovnania výsledkov v rovine úspešnosti vieme taktiež porovnať priestorové rozmiestnenie množín každej triedy. Je možné pozorovať, že pri nižších hodnotách k(1,3) sú pomyselné “hranice” tried ostrejšie menej pravidelné, pričom pri vyšších hodnotách k(15) je možno pozorovať pravý opak - hranice sú pomerne pravidelné.

Pri zvýšení hodnoty k zo 7 na 15 je taktiež možné pozorovať zmenu veľkosti “územia” najväčšej množiny tried.

Tento program bol riešený v jazyku Python, v IDE PyCharm.