Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Stefan Otto Novak, ID: 106078

ZADANIE 4A – KLASIFIKÁCIA

Umelá inteligencia

Študijný program: Informatika 4
Prednášajúci: Ing. Lukáš Kohútka, PhD.
Cvičiaci: Ing. Ivan Kapustík
december 2021

Obsah:

1. Zadanie úlohy

Zadanie 4a – klasifikácia

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 kvalifikátora 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.

2. Požívateľské rozhrania - ako používať program

Po spustení programu, požívateľ musí zvoli spôsob klasifikovania. Môže zvoliť klasifikovať body konkrétnej k-hodnotou alebo klasifikovania podľa testovacieho scenáru zo zadania (klasifikovať tie ešte body, ale s viacerými k-hodnotami, a to 1, 3, 7 a 15). Všetky zvolenia sa deje v konzole. Ak používateľ zvoli testovací scenár so zadania, ešte musí zadať počet generovaných bodov a následne program je spustený. Ak používateľ vzali sobom koketnej k-hodnoty, tak následne musí zadať k- hodnotu a počet generovaných bodov.

Správnosť výpočtov bude vypišaný v konzole a mapa bude vykreslená na Plote použitého IDE.

3. Riešenie problému a fungovania algoritmu

a) Generovanie štartovacích bodov

Po zvolení spôsobu klasifikácie, program nepotrebuje už žiadny input so strany používateľa a začne riešiť problém. Najprv vygeneruje začiatočne body (5 pre každú triedu), ktoré sú spomenuté v zadaní volaním funkcie ***generate_starting_dots()*.** Táto funkcia generuje inštancie triedy ***Dot*** pre každý bod so zadania a ich ukladá do dvoch polí, do póla ***DOTS***, kde sú zapísané body bez klasifikácie a do póla ***CLASSIFIED_DOTS***, kde následne budú zapísané klasifikovane body. V prípade štvoritého klasifikovania (scenár so zadania), body sú zapísané do zvlášť rôznych pól pre každú hodnotu klasifikovania.

```
def generate_starting_dots():
    starting_dots = [[-4500, -4400, "r"], [-4100, -3000, "r"], [-1800, -2400, "r"], [-2500, -3400, "r"], [-2000, -1400, "r"],
                     [4500, -4400, "g"], [4100, -3000, "g"], [1800, -2400, "g"], [2500, -3400, "g"], [2000, -1400, "g"],
                     [-4500, 4400, "b"], [-4100, 3000, "b"], [-1800, 2400, "b"], [-2500, 3400, "b"], [-2000, 1400, "b"],
                     [4500, 4400, "m"], [4100, 3000, "m"], [1800, 2400, "m"], [2500, 3400, "m"], [2000, 1400, "m"]]

    for s_dot in starting_dots:
        new_dot = Dot(s_dot[0], s_dot[1], s_dot[2])     # Vytvorim nove instancie startovacich bodo
        DOTS.append(new_dot)                            # a nasledne ich pridam do pola nekalsifikovancych

        if CLASSIFI_4METHOD == False:
            CLASSIFIED_DOTS.append(new_dot)
```

b) Generovania a klasifikácia bodov

Po úspešnou generovaní začiatočných bodov, nasleduje generovanie a klasifikácia nasledovných bodov. Pre rešpektovania podmienky so zadania – vždy nový bod bude mat zakaždým inú triedu, zvolil som generovanie bodov pomocou *for* cyklu, ktorý ide až po 25% celkového počtu bodov a pri každej iterácii budú generované body pre každú triedu.

```
def generate_dots(n_dots):
    for i in range(int(n_dots / 4)):
        generate_colored_dot(-5000, 0, -5000, 0, "r")
        generate_colored_dot(-5000, 0, 0, 5000, "b")
        generate_colored_dot(0, 5000, -5000, 0, "g")
        generate_colored_dot(0, 5000, 0, 5000, "m")
```

Konkrétne generovanie inštancii bodov sa sa deje vo funkcii ***generate_colored_dot***. Táto funkcia je volaná v funkciou ***generate_dots*** a prijíma ako parametre intervaly x a y osi a farbu bodu. Táto funkcia generuje náhodné hodnoty pre body z intervalu a následne vytvorí inštanciu bodu. Funkcia implementuje aj podmienku zadania, ktorá hovorí o 1% šance generovania bodu s rôznou farbou ako štandardná farba kvadrant kde sa nachádza a zároveň aj podmienku generovania bodov s rôznymi súradnicovými. Po splnení týchto podmienkach, bod je zapísaný do póla ***DOTS***.

Kópie bodov ktoré sú pridane do póla ***DOTS***, sú klasifikované a pridané do póla ***CLASSIFIED_DOTS***. Ak trieda bodu po klasifikácie je rozdielná od štandardného generovania, tak potom je evidovaný ako rozdielna trieda do póla ***DIFERENT_COLOR***, ktorá slúži pre výpočet správnosti klasifikácie na konci programu. Rovnako sa deje aj v prípade keď program bude klasifikovať v tom istom čase pre viac k-hodnoty, ale klasifikovane body sú zapísané do zvlášť rôznych pól pre každú k-hodnotu.

```
clsf_dot = classify(dot, K, CLASSIFIED_DOTS)
if dot.color != clsf_dot.color:
    DIFERENT_COLOR += 1
CLASSIFIED_DOTS.append(clsf_dot)
```

c) Klasifikácia bodov

Klasifikácia bodov je uskutočnená pomocou funkcie ***classify()***, táto funkcia prijíma ako parametre: bod, ktorý bude klasifikovaný, k-hodnotu a pole už klasifikovaných bodov. Táto funkcia vráti novo vytvorenú inštanciu bodu, ktorá bude mať rovnaké súradnice ako bod ktorý prijímala, ale farba bodu bode rozhodnutá na základe susedov. Počet susedov ktorý budú brány do úvahy je reprezentovaný *k*-hodnotou. Klasifikovaný bod dostane takú farbu, aká dominuje v oblasti k-najbližších susedov. Všetky štyri farby sú zapísané do jednej knižnice, do ktorej je zapísaná evidencia farby susedných bodov. Následne, knižnica je usporiadaná pomocou funkcie *sort a* finálna klasifikujúca farba je nastavená na základe prvej farby v knižnice. Konkrétny vyber najbližších susedov je uskutočnení pomocou funkcie ***k_NN().***

```
def classify(dot, k: int, dots):
    red = green = blue = magenta = 0
    neighbors = k_NN(dot, k, dots)
    for neighbor in neighbors:
        if (neighbor[0].color) == "r":
            red += 1
        if (neighbor[0].color) == "g":
            green += 1
        if (neighbor[0].color) == "b":
            blue += 1
        if (neighbor[0].color) == "m":
            magenta += 1
    colors = {"r": red, "g": green, "b": blue, "m": magenta}
    color = max(colors, key=colors.get)

    new_dot = Dot(dot.x, dot.y, color)
    return new_dot
```

d) Výber susedov

Výber susedov vykonáva funkcia ***k_NN()***, ktorá jednoducho prejde cez všetky body a na základe vzdialenosti medzi bodmi, vráti k najbližšie body. Vzdialenosť medzi bodmi je vypočítaná pomocou funkcie ***calculate_distance()*.**

```
for clsf_dot in dots:
    distance = calculate_distance(dot, clsf_dot)
    dist.append([clsf_dot, distance])

dist.sort(key=operator.itemgetter(1))
for i in range(k):
    neighbors.append(dist[i])
```

e) Výpočet vzdialenosti medzi bodmi

Výpočet vzdialenosti medzi bodmi je uskutočnený pomocou funkcie ***calcualte_distance()*.** Funkcia vypočítať vzdialenosť medzi dvomi bodmi na mape pomocou Euklidovského vzorca a vracia iba túto hodnotu.

```
def calculate_distance(dot1, dot2):
    distance = 0
    distance += (dot1.x - dot2.x) ** 2
    distance += (dot1.y - dot2.y) ** 2
    distance = int(sqrt(distance))

    return distance
```

Po uskutočnení týchto všetkých krokov, program sa vracia späť do ***main()*** funkcie, kde už iba vypisuje správnosť programu a vykresli mapy pomocou funkcie ***dispaly_plot()***. Správnosť programu je vypočítaná percentuálne na základe celkového počtu bodov a bodov ktoré si zmenili triedy po klasifikácie, táto hodnota je vypísaná v konzole.

```
display_plot(CLASSIFIED_DOTS, K, n_dots)
    accuracy = 100 * float(-1 * (DIFERENT_COLOR - n_dots))/float(n_dots)
    print("Accuracy of classifier:", accuracy, "%")
```

f) Výkres mapy

Výkres mapy je uskutočnený funkciou ***display_plot()***, ktorá prijíma ak parametre body, *k* hodnotu a počet všetkých bodov. Konkrétny výpis mapy je uskutočnený pomocou knižnice *matplotlib.* Z tejto knižnice sú použite viac funkcie ako napr. *scatter(), title(), grid()* a *show()*. Funkcia *scatter()* vykresli bod na mape na základe x hodnoty, y hodnoty a farby bodu. Táto funkcia je volaná pre každý pod so zoznamu bodov. *title()* pridá nadpis vypíšu, *grip()* pridá orientačne čiary a *show()* vykresli celú mapu.

```python
def display_plot(dots, k, n_dots):
    for dot in dots:
        plt.scatter(dot.x, dot.y, color=dot.color)
    if k == 0:
        title = "Without classification." + str(n_dots)+ " - dots: " + str(n_dots)
    else:
        title = "With classification. k: " + str(k) + " - dots: " + str(n_dots)
    plt.title(title)
    plt.grid()
    plt.show()
```

4. Testovanie a dosiahnuté výsledky

Program bol testovaný podľa testovacieho scenára (experiment) zo zadina, ale aj iné menšie scenáre.

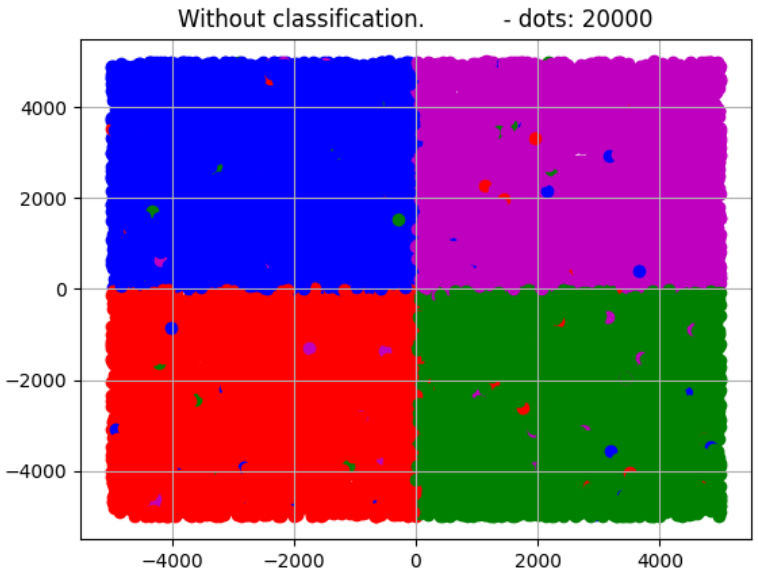
Program bol testovaný v programovacom prostredí Pychar a program bol písaný v programovacom jazyku Python 3.0. Dole uvedene výsledný boli dosiahnuté so zaradením s nasledovnými špecifikáciami: procesor i5 7300HQ, 16 GB Ram DDR4 a 512GB SSD.

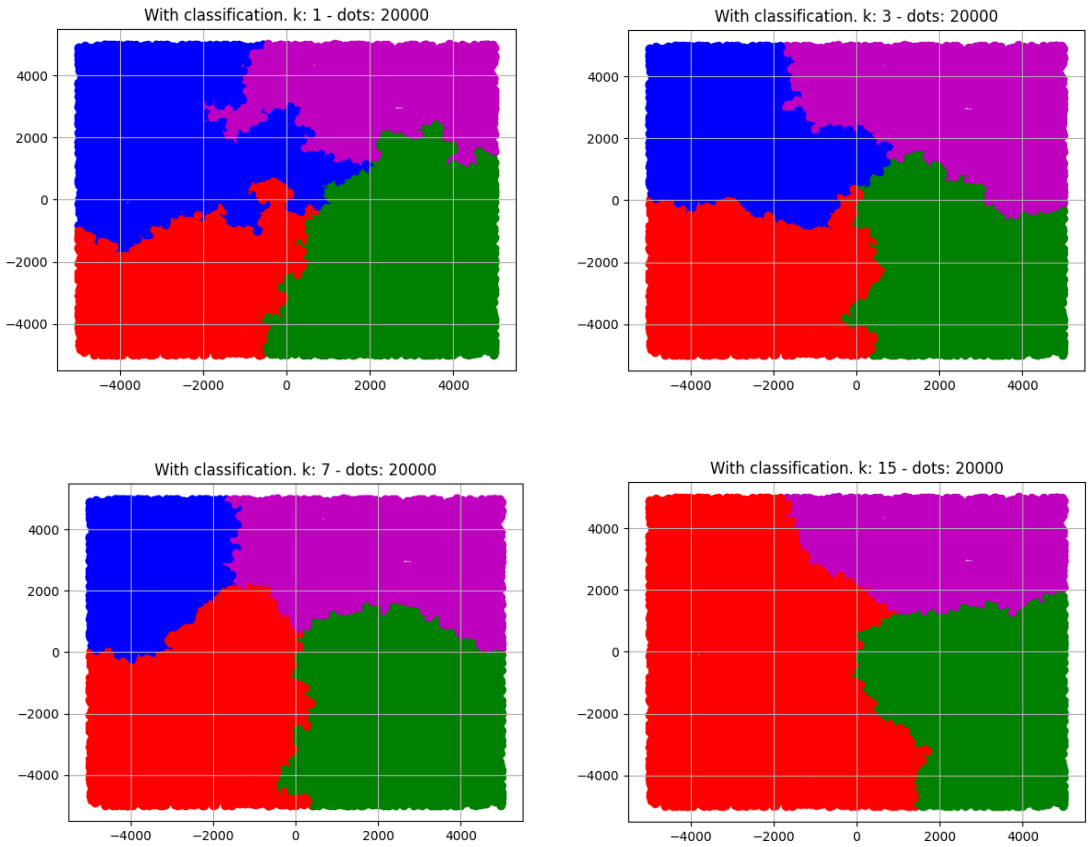
a) Experiment so zadania – spustenie programu 4-krát, pričom zakaždým kvalifikátor použije iný parameter k (pre k = 1, 3, 7 alebo 15) a vygenerované body budú pre každý experiment rovnaké. Tento scenár spočíva z 20.000 generovaných bodov. Dokončenie tohto scenáru trhávalo **1h a 34 minút.**

Dosiahnuté výsledky:

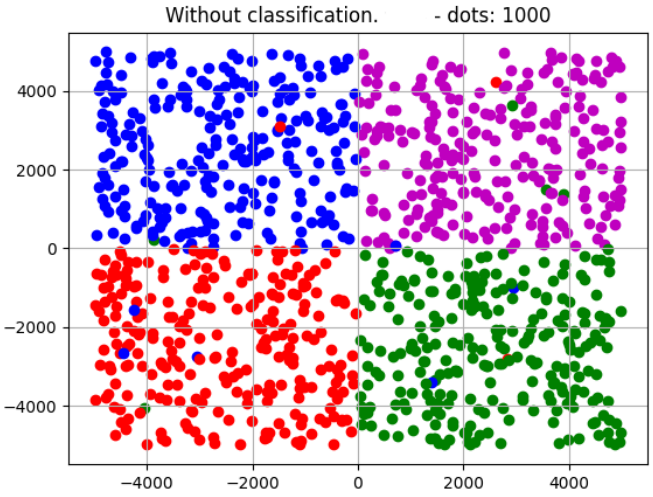
- Pre k = 1 - 81.18%
- Pre k = 3 – 87.31%
- Pre k = 7 – 83.55%
- Pre k = 15 – 62.90%

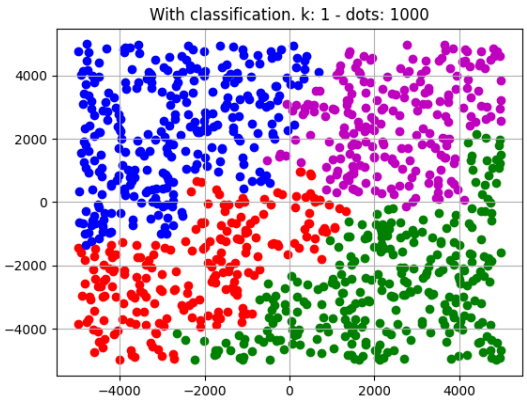
```
Insert the number of dots: 20000
Accuracy of classifier k = 1: 81.185 %
Accuracy of classifier k = 3: 87.31 %
Accuracy of classifier k = 7: 83.555 %
Accuracy of classifier k = 15: 62.905 %
```

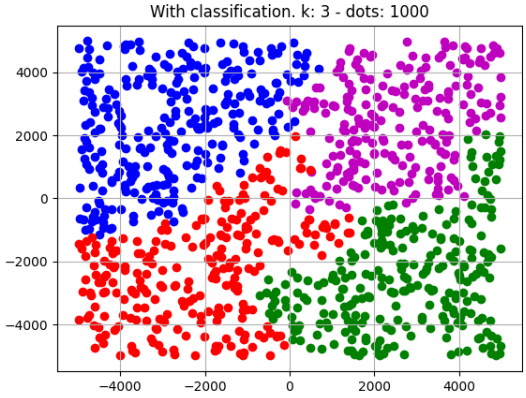


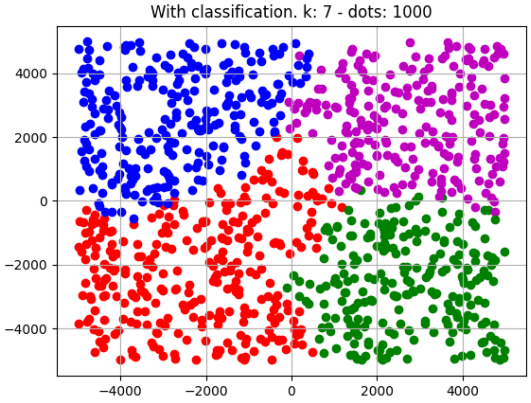


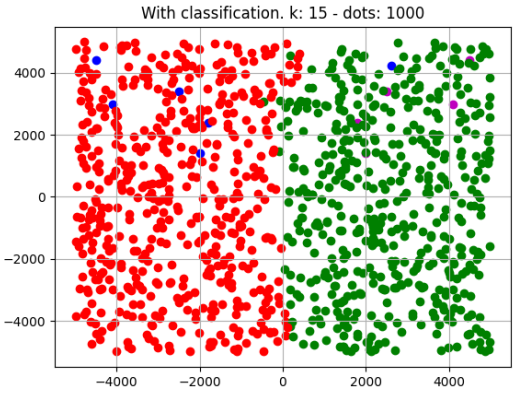
a) Generovanie podobného scenáru ako scenár **a),** ale s 1.000 generovanými bodmi. Dokončenie tohto scenáru trhávalo **36 sekúnd.**



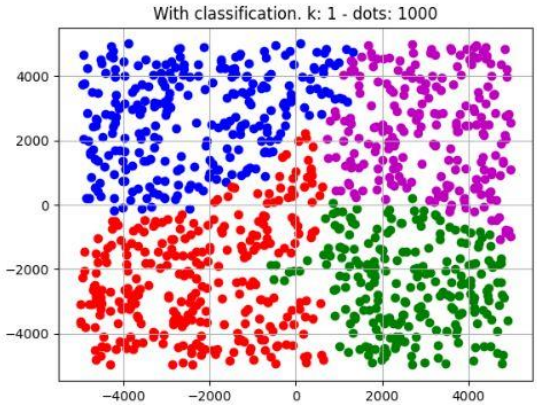


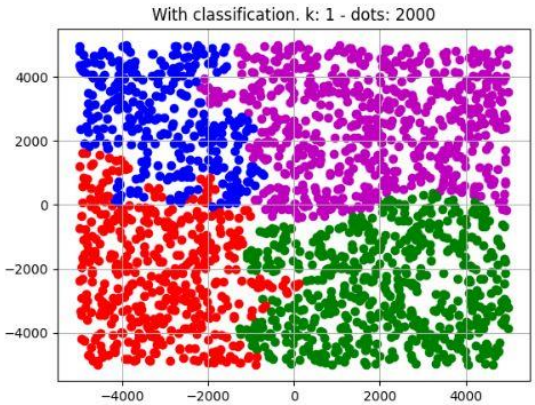


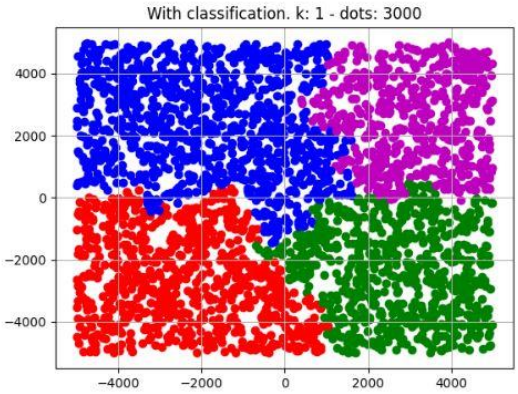


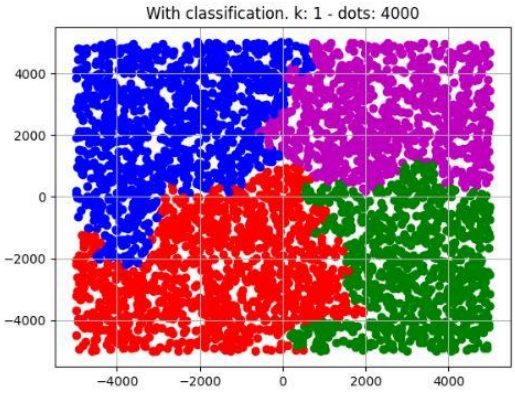


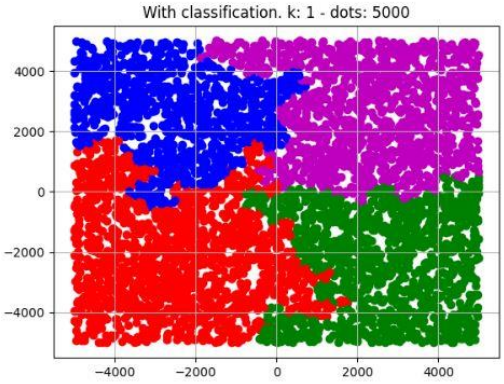
b) Porovnanie časov a úspešnosti pri k-hodnote = 1 pre rôzne hodnoty generovaných bodov (1000, 2000, 3000, 4000 a 5000):



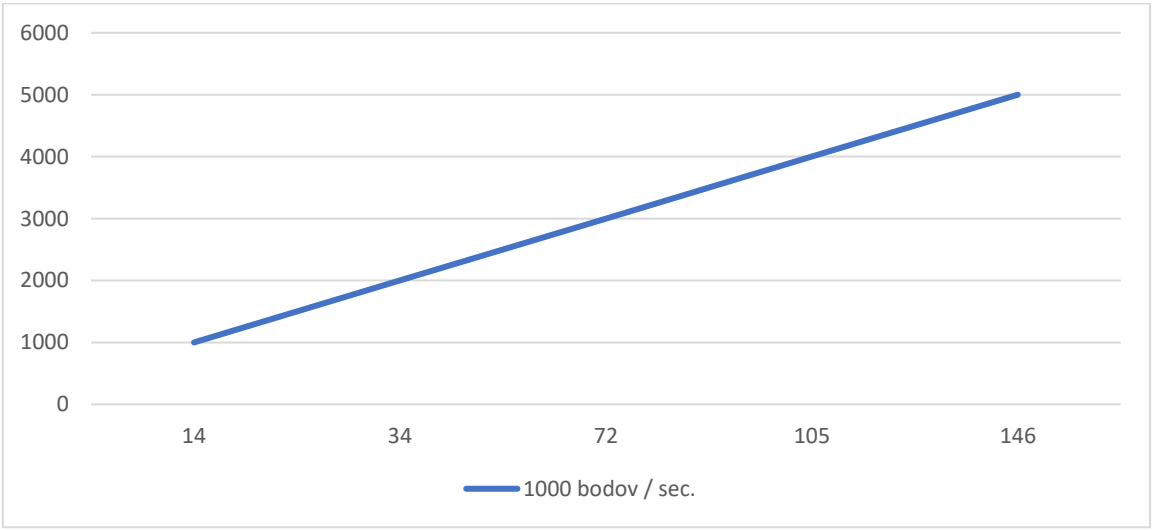




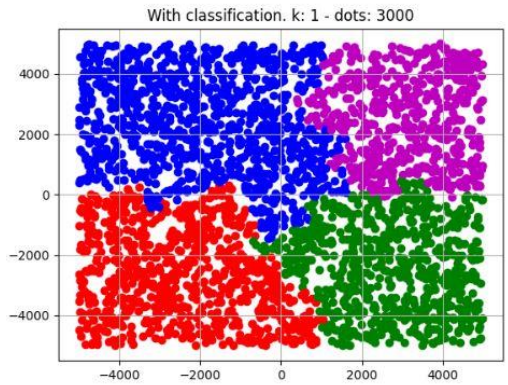


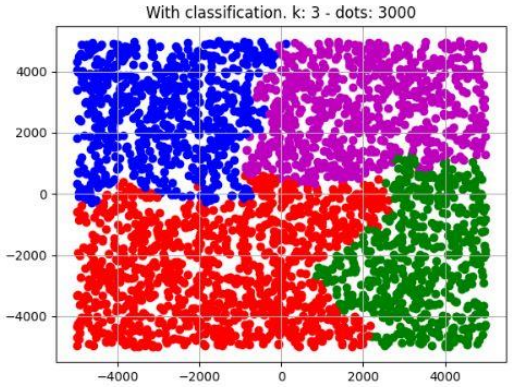


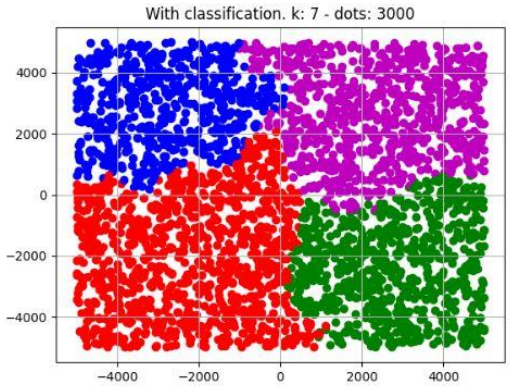
Počet bodov	Čas vykonania	Úspešnosť
1000	14 sec.	88%
2000	34 sec.	83%
3000	1 min. 12 sec.	86%
4000	1 min. 45 sec.	84%
5000	2 min. 26 sec.	87%

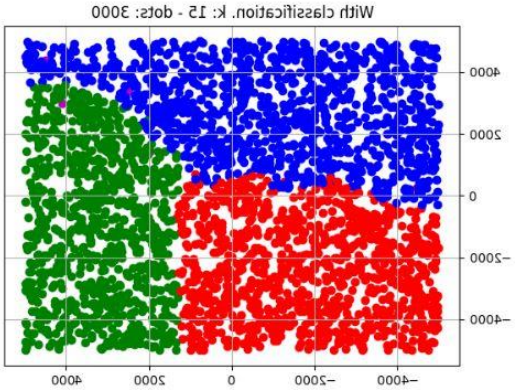


c) Porovnanie časov a úspešnosti pri rôznych k-hodnôt (1, 3, 7 a 15) pre rovnaký počet bodov (3000 v tomto konkrétno prípade):

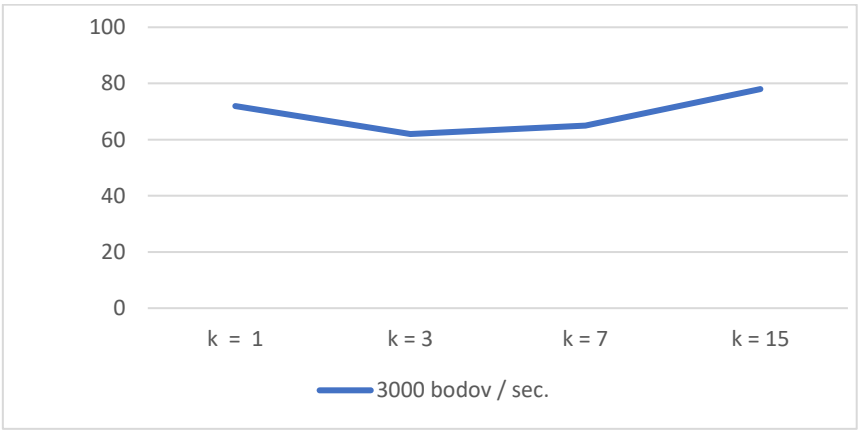








Počet bodov	Čas vykonania	Úspešnosť
3000	1 min 12 sec.	86%
3000	1 min. 2 sec.	82%
3000	1 min. 5 sec.	88,5%
3000	1 min. 8 sec.	67%



Na základe testovacích scenárov a výsledkov sú potvrdené nasledovné tvrdenia:

- Čim k-hodnota je väčšia, tým hranica medzi farbami je “rovnejšia”.
- Pri k = 15 môžeme vidieť fenomén skoro kupletného zániku jednej alebo dvoch farieb. Je to z dôvodu malého počtu stravovacích bodov.
- Keď k- hodnota je rovnaká tak, čas vykonania výpočtu závisí (skoro lineárne) v súlade s počtom generovaných bodov. Neovplyvni výrazne ani úspešnosť. Iba pre k-hodnotách porovnateľných s počtom štartovacích bodov.
- K-hodnota neovplyvni výrazne čas vykonania pri rovnakom pocte bodov a neovplyvni výrazne ani úspešnosť. Iba pre k-hodnotách porovnateľných s počtom štartovacích bodov.