Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies

Umelá inteligencia

Zadanie 3 - klasifikácia

Petra Miková

Petra Miková
ID: 120852

Obsah

Petra Miková
ID: 120852

Opis riešenia

Mojou úlohou bolo naprogramovať klasifikátor pre nové body pomocou funkcie classify, kde x a y sú súradnice nového bodu. Klasifikátor mal používať k-NN algoritmus s hodnotami k rovnými 1, 3, 7 alebo 15. Po klasifikácii mal byť nový bod pridaný do 2D priestoru s priradenou farbou podľa jeho klasifikácie.

K-NN algoritmus

K-NN (k-Nearest Neighbors) je algoritmus používaný pre klasifikáciu a regresiu. Pri klasifikácii nového bodu určuje jeho triedu na základe triedy jeho najbližších susedov zo vstupného priestoru. Parametrom k sa určuje, koľko najbližších susedov sa má zohľadniť. V mojej implementácii som ešte využila algoritmus na hľadanie práve k najmenších hodnôt kvôli optimalizácii riešenia.

Reprezentácia údajov

Na reprezentáciu údajov jedného bodu používam triedu Point, ktorá vyzerá nasledovne:

```
class Point:
    def __init__(self, x, y, color):
        self.x = x
        self.y = y
        self.color = color
```

Udržiava v sebe informáciu o x a y súradniciach, a taktiež o farbe.

Funkcia classify

Hlavná funkcia classify implementuje klasifikáciu nového bodu v 2D priestore pomocou k-NN algoritmu. Vstupné parametre sú súradnice nového bodu (X, Y) a hodnota k, ktorá určuje počet najbližších susedov, ktoré sa berú do úvahy pri klasifikácii.

```
def classify(x, y, k):
    global training_data
    global classified_data

    min_distances = heapq.nsmallest(k, ((euclidean_distance(Point(x, y,
""), point), point.color) for point in
                                        training_data))

    color_counts = {'red': 0, 'green': 0, 'blue': 0, 'purple': 0}
    for distance, color in min_distances:
        color_counts[color] += 1

    new_color = max(color_counts, key=color_counts.get)
    classified_data.append(Point(x, y, new_color))
    training_data.append(Point(x, y, new_color))

    return new_color
```

Funkcia využíva optimalizačnú metódu pomocou algoritmu hľadania k najmenších hodnôt. Pre nový bod sa vypočíta Euklidovská vzdialenosť ku všetkým bodom v training množine, a následne sa vyberie k najbližších susedov s priradenými farbami. Početnosti farieb medzi najbližšími susedmi určujú pravdepodobnú farbu nového bodu. Nový bod s priradenou farbou sa následne pridá do klasifikovaných dát a do training množiny. Návratovou hodnotou funkcie je farba, ktorú dostal nový bod po klasifikácii.

Petra Miková
ID: 120852

Euklidovskú vzdialenosť počítam v pomocnej funkcii:

```
def euclidean_distance(point1, point2):
    return math.sqrt((point2.x - point1.x)**2 + (point2.y - point1.y)**2)
```

Generovanie bodov

Prvotných 20 bodov generujem veľmi jednoducho, keďže ich poloha je vopred daná:

```
starting_points = {
    'red': [(-4500, -4400), (-4100, -3000), (-1800, -2400), (-2500, -3400),
(-2000, -1400)],
    'green': [(4500, -4400), (4100, -3000), (1800, -2400), (2500, -3400),
(2000, -1400)],
    'blue': [(-4500, 4400), (-4100, 3000), (-1800, 2400), (-2500, 3400), (-
2000, 1400)],
    'purple': [(4500, 4400), (4100, 3000), (1800, 2400), (2500, 3400),
(2000, 1400)]
}
```

Zvyšné body podľa podmienok generujem vo funkcii generate_points:

```
def generate_points(number_of_points_per_color):
    colors = ['red', 'green', 'blue', 'purple']
    global red
    global green
    global blue
    global purple

    for color in colors:
        for _ in range(number_of_points_per_color):
            if color == 'red': #Červená
                if random.randint(1, 100) > 1:
                    x = random.randint(-5000, 500)
                    y = random.randint(-5000, 500)
                else:
                    while x < 500 and y < 500:
                        x = random.randint(-5000, 5000)
                        y = random.randint(-5000, 5000)

                point = Point(x, y, color)
                if point not in red:
                    unique_point = point
                red.append(unique_point)
            elif color == 'green': #Zelená
                if random.randint(1, 100) > 1:
                    x = random.randint(-500, 5000)
                    y = random.randint(-5000, 500)
                else:
                    while x > -500 and y < 500:
                        x = random.randint(-5000, 5000)
                        y = random.randint(-5000, 5000)

                point = Point(x, y, color)
                if point not in green:
                    unique_point = point
                green.append(unique_point)
            elif color == 'blue': #Modrá
                if random.randint(1, 100) > 1:
                    x = random.randint(-5000, 500)
```

```
                    y = random.randint(-500, 5000)
                else:
                    while x < 500 and y > -500:
                        x = random.randint(-5000, 5000)
                        y = random.randint(-5000, 5000)

                point = Point(x, y, color)
                if point not in blue:
                    unique_point = point
                blue.append(unique_point)
            else: #Fialová
                if random.randint(1, 100) > 1:
                    x = random.randint(-500, 5000)
                    y = random.randint(-500, 5000)
                else:
                    while x > -500 and y > -500:
                        x = random.randint(-5000, 5000)
                        y = random.randint(-5000, 5000)

                point = Point(x, y, color)
                if point not in purple:
                    unique_point = point
                purple.append(unique_point)
```

Pre každú farbu sa generuje určený počet bodov s náhodnými súradnicami (X, Y) v požadovanom rozsahu podľa pravdepodobnosti, ktorá je náhodná. Generované body sa ukladajú do príslušných zoznamov (red, green, blue, purple) s kontrolou duplicít. Každý vygenerovaný bod je vytvorený ako inštancia triedy Point a následne pridaný do príslušného zoznamu pre farbu danej inštancie.

Funkcia classify_point

Funkcia classify_point je určená na klasifikáciu bodu z určenej farby a následné sledovanie chýb v klasifikácii.

```
def classify_point(color, color_count, color_list, color_index, k):
    global errors

    if color_count < len(color_list[color_index]):
        point = color_list[color_index][color_count]
        x, y = point.x, point.y
        color_count += 1

        new_color = classify(x, y, k)

        if new_color != color:
            errors += 1

    return color_count
```

Volá sa v main loope pre každý bod zo zoznamu a vykonáva na ňom classify funkciu.

Vykreslenie výsledkov

Na vykreslenie používam tkinter, pôvodne som skúšala matplotlib, ale ten nebol kompatibilný s PyPy.

```
def plot_points(classified_data):
    root = tk.Tk()
    canvas_width = 800
    canvas_height = 800
    canvas = tk.Canvas(root, width=canvas_width, height=canvas_height)
    canvas.pack()

    min_x = min(point.x for point in classified_data)
```

```
    max_x = max(point.x for point in classified_data)
    min_y = min(point.y for point in classified_data)
    max_y = max(point.y for point in classified_data)

    scale_factor_x = canvas_width / (max_x - min_x)
    scale_factor_y = canvas_height / (max_y - min_y)

    for point in classified_data:
        x, y, color = point.x, point.y, point.color

        x_scaled = (x - min_x) * scale_factor_x
        y_scaled = (y - min_y) * scale_factor_y

        oval_size = 20
        canvas.create_oval(x_scaled - oval_size, y_scaled - oval_size,
x_scaled + oval_size, y_scaled + oval_size,
                           fill=color, outline=color)

    root.mainloop()
```

Funkcia vytvorí plátno a scale factor pre body, aby sa to korektne vykreslilo. Neskôr už iba vo for loope priamo vytvorí reprezentáciu každého bodu cez create oval.

Main

Tu sa vykonáva celý proces experimentu. Najprv sa zadá požadovaný počet bodov z každej farby a vygenerujú sa body, ktoré sa používajú pre každé k.

```
if __name__ == "__main__":
    number = 10000
    generate_points(number)

    for i in range(4):
        print("Experiment number ", i+1,":")
        k_values = [1, 3, 7, 15]
        for k in k_values:
            print("*------------*")
            print("|            |")
            print("|   k = ",k,"  |")
            print("|            |")
            print("*------------*")
            errors = 0
            training_data = [Point(x, y, color) for color, coordinates in
starting_points.items() for x, y in coordinates]
            classified_data = [Point(x, y, color) for color, coordinates in
starting_points.items() for x, y in coordinates]

            color_counts = {'red': 0, 'green': 0, 'blue': 0, 'purple': 0}
            color_list = {'red': red, 'green': green, 'blue': blue,
'purple': purple}

            while not all(count == number for count in
color_counts.values()):
                color = random.choice(['red', 'green', 'blue', 'purple'])
                color_counts[color] = classify_point(color,
color_counts[color], color_list, color, k)

            print("Errors for k =", k, ":", errors)
            print("Success rate: ", round(float(((number * 4 - errors) /
(number * 4)) * 100), 2), "%")
            plot_points(classified_data)
```

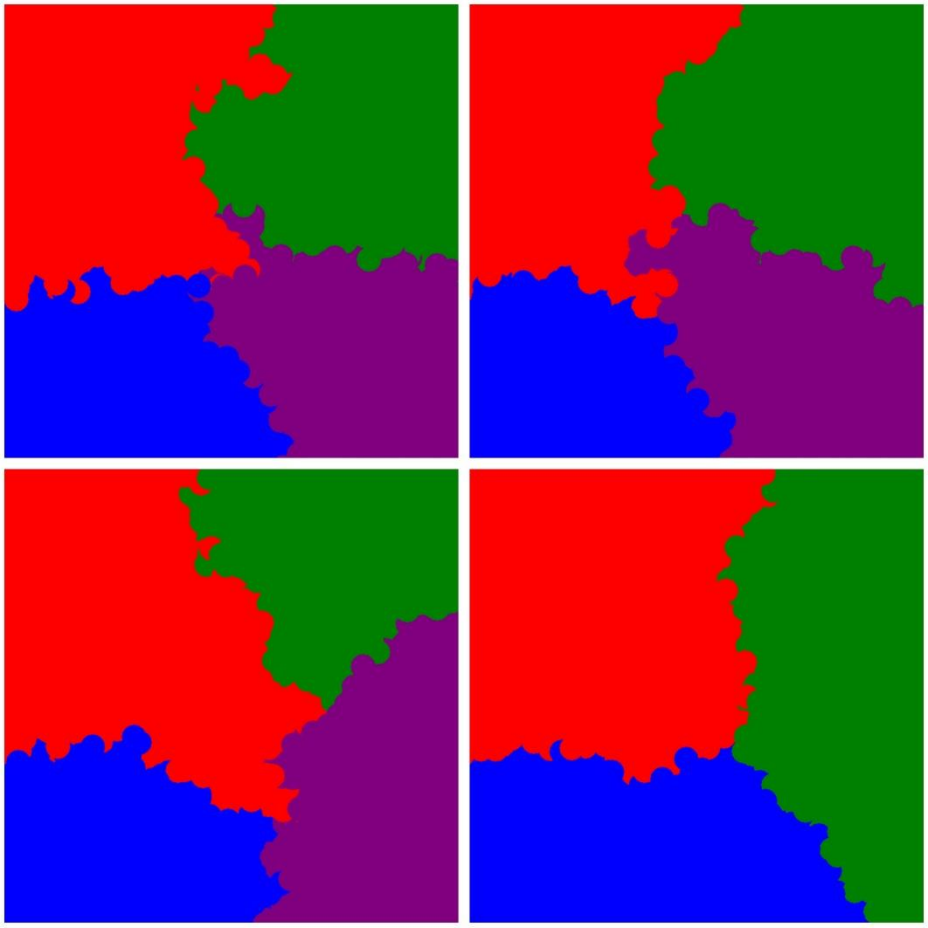
Petra Miková
ID: 120852

Na každom k sa vykonáva experiment spôsobom, že najprv si do training aj konečnej množiny pridám tie prvotné vygenerované body ktorých bolo 20, a potom postupne dokým neprejdem všetky farby volám na body funkciu classify_point, kde sa každý bod klasifikuje. Pre každý experiment na konci vypíšem počet chýb a success rate. Eventuálne sa experiment vykreslí do canvas.

Výsledok

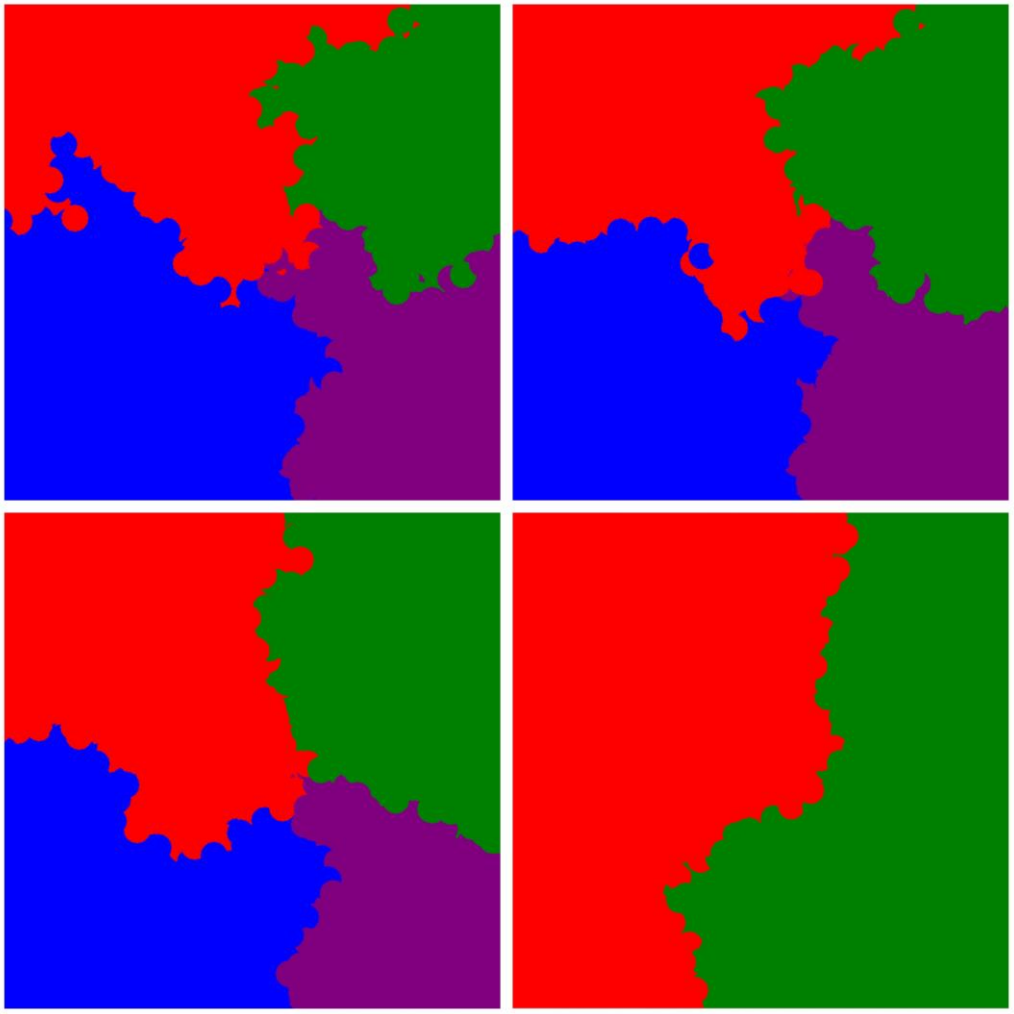
Experiment pre všetky k hodnoty som pustila 4 krát a toto boli výsledky, ktoré sú zoradené takto:
1 3
7 15

Pokus 1



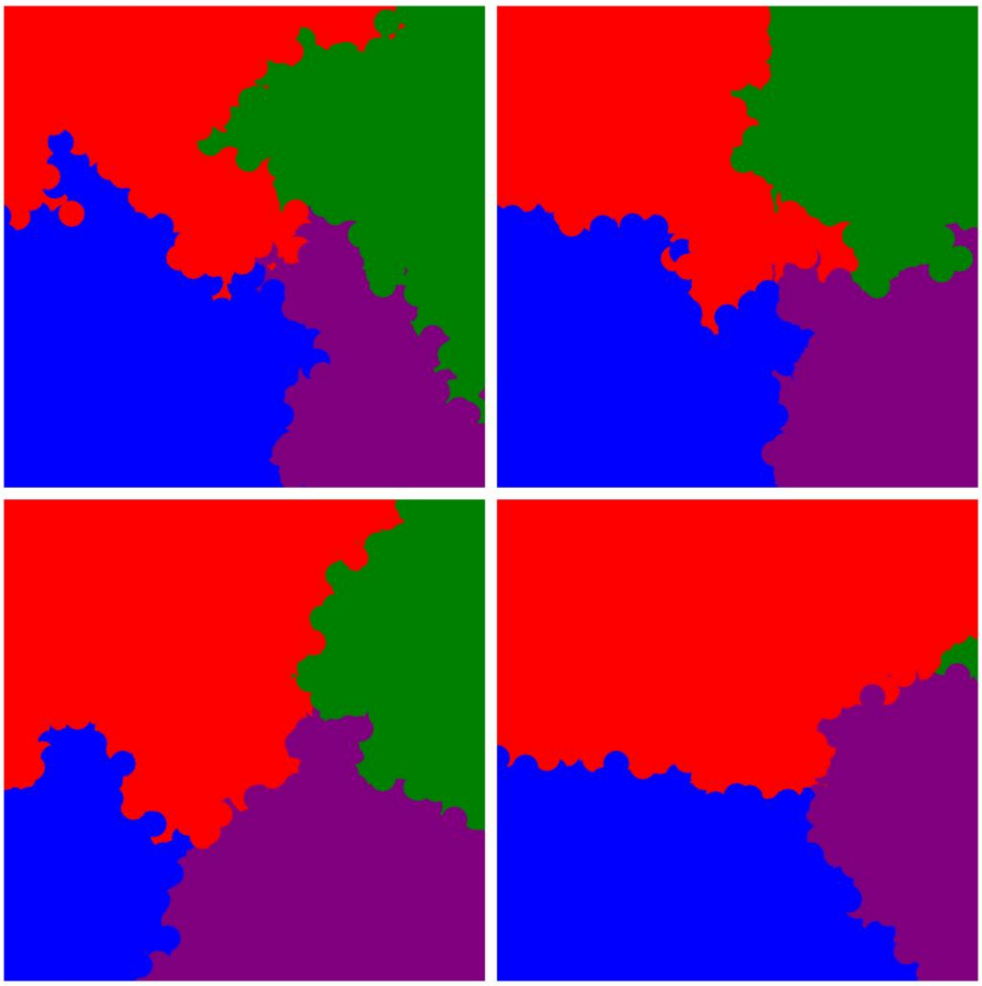
Petra Miková
ID: 120852

Pokus 2



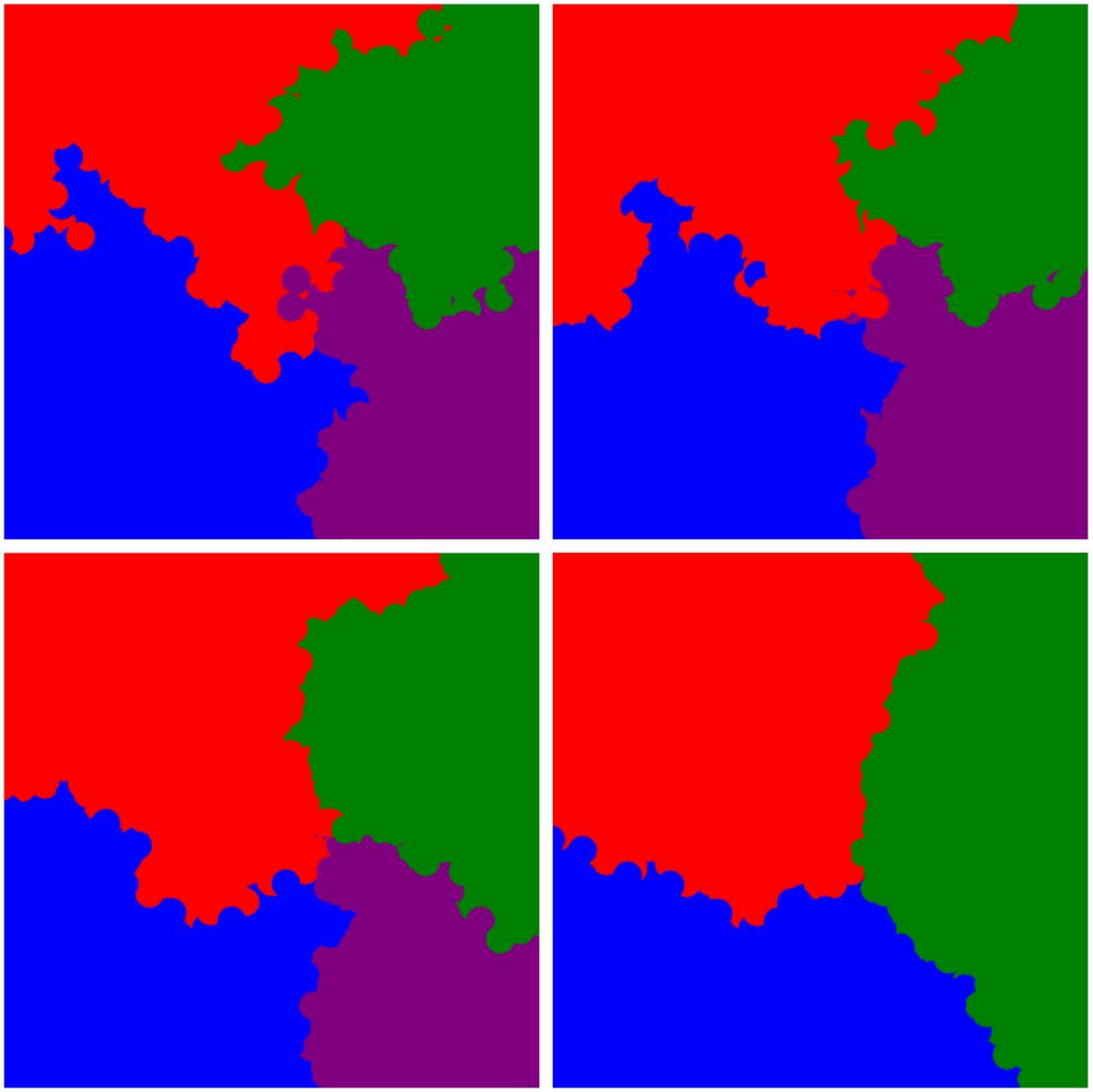
Petra Miková
ID: 120852

Pokus 3



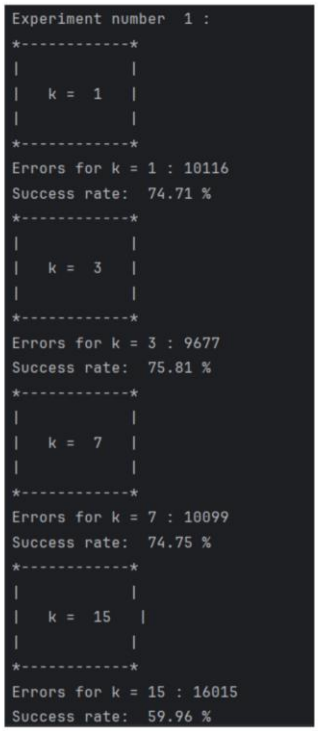
Petra Miková
ID: 120852

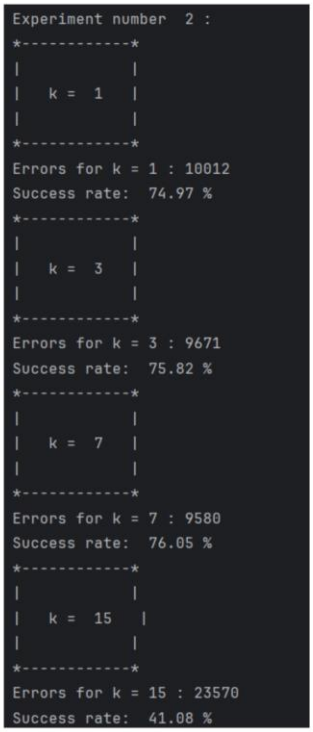
Pokus 4

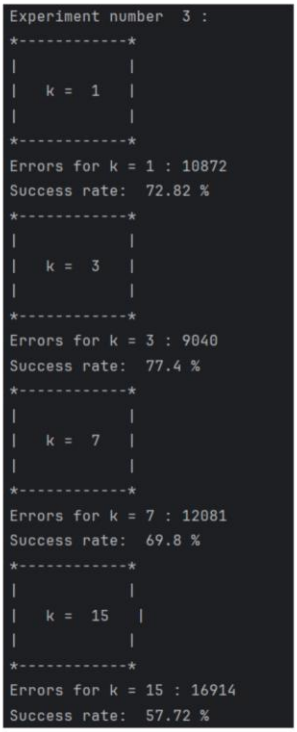


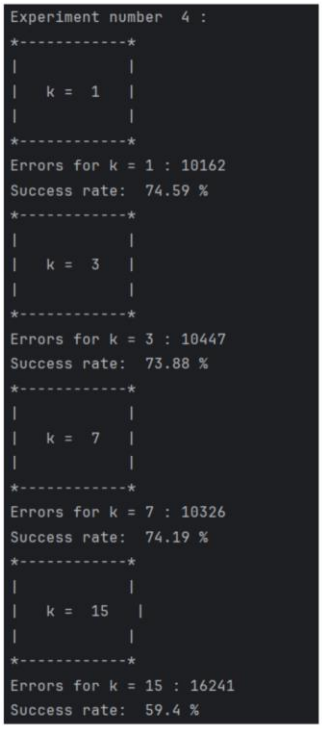
Petra Miková
ID: 120852

Výsledky pokusov









Záver

Zadaním bolo implementovať klasifikáciu bodov v 2D priestore pomocou K-NN algoritmu, čo sa mi aj podarilo splniť a svoju implementáciu som otestovala na hodnotách k= 1,3,7,15 hneď 4x, čo mi poskytlo spokojnosť s mojou implementáciou. Program som implementovala v jazyku Python a spúšťala kvôli urýchleniu pomocou PyPy, a taktiež som funkciu classify optimalizovala, keďže bez tohto všetkého mi to na 40 000 bodoch bežalo až niekoľko desiatok minút.