Systemy sztucznej inteligencji

dokumentacja projektu „Rozpoznawanie cyfr”

Mateusz Białecki, grupa 3E

26 maja 2021

Część I

Opis programu

Program ma na celu wykrycie jaka cyfra została narysowana na matrycy 8 na 8 pikseli, przy pomocy algorytmu k-najbliższych sąsiadów.

Instrukcja obsługi

Zaleca się korzystanie z programu w środowisku Jupyter poprzez zaimportowanie do niego zeszytu „Digit_recognition.ipynb" i następnie wykonywanie kolejnych komórek zeszytu (Shift + Enter).
Ważne, aby w tym samym folderze co skrypt, znalazła się baza o nazwie „digit_matrices.csv". Dane w ww. bazie to kolejne rekordy będące reprezentacjami matryc 8x8, na których narysowano cyfry. Każdy rekord składa się z 65 pól. Pola 1 - 64 reprezentują wartości kolejnych pikseli (idąc od lewego górnego rogu w prawo, następnie w dół) w skali „intensywnośc" od 1 do 16. Ostatnie, 65. pole, identyfikuje cyfrę jaka została zapisana na matrycy.

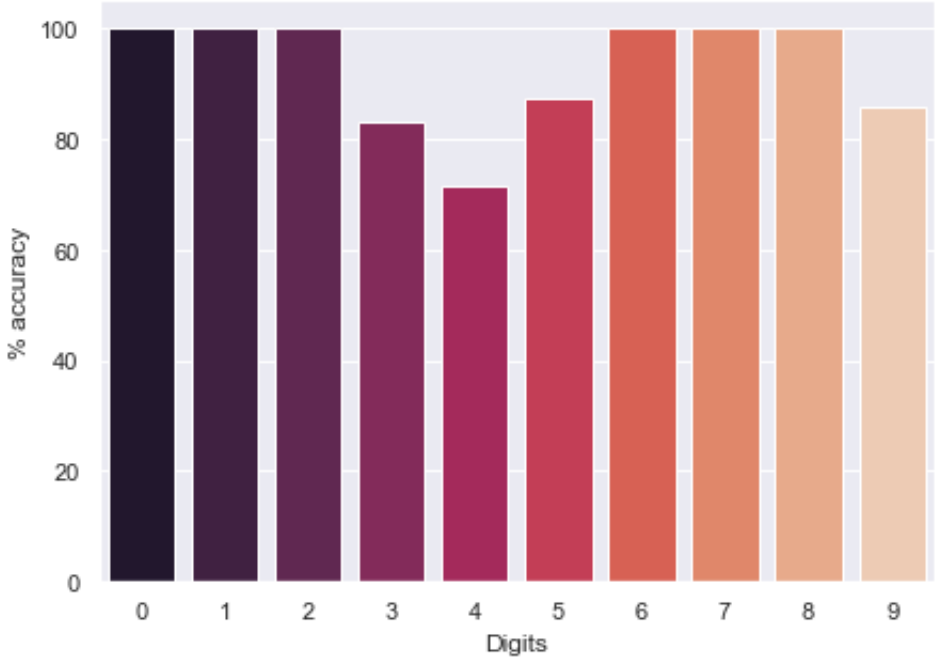
```
[86]: all_digits
```

[86]:

	1	2	3	4	5	6	7	8	9	10	...	56	57	58	59	60	61	62	63	64	Digit
0	0	0	10	16	16	10	0	0	0	4	...	0	0	0	13	15	11	8	14	7	2
1	0	0	5	13	7	0	0	0	0	0	...	3	0	0	4	8	10	16	12	1	2
2	0	0	0	10	16	11	1	0	0	0	...	0	0	0	0	8	15	1	0	0	1
3	0	0	13	16	15	2	0	0	0	5	...	0	0	0	15	7	0	0	0	0	7
4	0	0	1	10	14	13	1	0	0	0	...	0	0	0	0	12	14	4	0	0	5

Każda odpowiedź algorytmu jest sprawdzana z poprawną odpowiedzią. Jeżeli różnią się, drukowany jest komunikat o tym, którą cyfrę pomylono z którą. Po wykonaniu wszystkich obliczeń wyświetlany jest wykres dokładności dla każdej cyfry.

```
TESTING WITH k=2 and m=2

Wrong prediction. Digit: 3, guessed: 2
Wrong prediction. Digit: 4, guessed: 0
Wrong prediction. Digit: 5, guessed: 3
Wrong prediction. Digit: 4, guessed: 0
Wrong prediction. Digit: 9, guessed: 3
Accuracy: 91.66666666666667%
```



Dodatkowe informacje

Wymagania

- Python 3 (testowano na 3.9.1 64-bit)
- biblioteki *numpy, pandas, matplotlib, seaborn*
- środowisko Jupyter

Część II

Opis działania

Program dzieli zadany zbiór na część walidacyjną i testową w proporcji 70:30. Następnie bierze po jednej próbce z części testowej, sprawdza jej odległość (korzystając z metryki Minkowskiego) do każdej z próbek w zbiorze waldiacyjnym, a ostatecznie wybiera k najbliższych próbek (czyli takich o najmniejszej odległości do sprawdzanej próbki testowej), które przeprowadzają „głosowanie". Zwracana jest cyfra która uzyskała najwięcej głosów.
Chcąc zilustrować to na przykładzie uprośćmy całą sytuację dla ułatwienia obliczeń.
Załóżmy, że w zbiorze treningowym znajduje się sześć próbek, które mają po 4 pola. Chcemy sprawdzić, czy naszej próbce jest bliżej do 0 czy do 1.

```
probka
```

	1	2	3
0	1	2	3

```
zbior_walidacyjny
```

	1	2	3	Digit
0	1	2	3	1
1	3	2	4	1
2	1	4	1	0
3	2	2	2	1
4	1	3	4	0
5	4	1	1	0

Dystans próbki testowej do próbki ze zbioru liczymy jako pierwiastek m-tego stopnia z sumy odległości do potęgi m odpowiadających sobie pól obu próbek.

$$d = \sqrt[m]{\sum_{i=1}^{n} |t_i - w_i|^m}$$

Gdzie n - ilość pól próbki testowej, t_i - wartość kolejnego pola próbki testowej, w_i - wartość kolejnego pola próbki walidacyjnej.
Załóżmy że werdykt chcemy oprzeć o 3 najbliższych sąsiadów (k = 3) i niech m = 1 (metryka Manhattan).
Dystans próbki testowej do próbki walidacyjnej nr 1:

$$|1-1| + |2-2| + |3-3| = 0$$

Dystans do próbki nr 2:

$$|1-3| + |2-2| + |3-4| = 3$$

Dystans do próbki nr 3:

$$|1-1|+|2-4|+|3-1|=4$$

Dystans do próbki nr 4:

$$|1-2|+|2-2|+|3-2|=2$$

Dystans do próbki nr 5:

$$|1-1|+|2-3|+|3-4|=2$$

Dystans do próbki nr 6:

$$|1-4|+|2-1|+|3-1|=6$$

Jak widać najmniejsze odległości uzyskaliśmy dla pierwszej, czwartej i piątej próbki. Próbki te wskazują kolejno 1, 1, 0, stąd większością głosów zwycięża 1 - i taką właśnie cyfrę typuje algorytm.

```
zbior_walidacyjny
```

	1	2	3	Digit
0	1	2	3	1
1	3	2	4	1
2	1	4	1	0
3	2	2	2	1
4	1	3	4	0
5	4	1	1	0

Algorytm

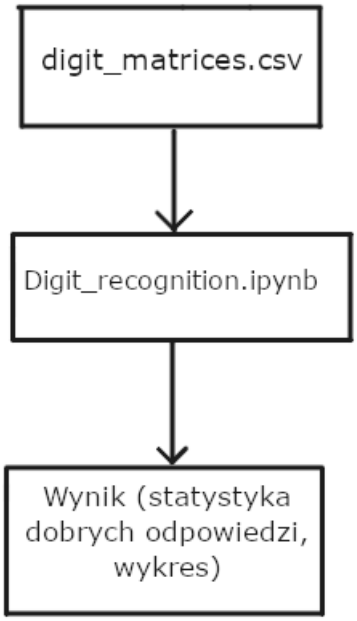
Pseudokod skryptu:

Data: Dane wejściowe baza danych liczb *all_digits*
Result: Brak
Wyświelt kilka statystyk dotyczących poszczególnych pikseli matryc;
Przetasuj bazę;
Podziel zbiór na dwa zbiory, walidacyjy oraz testowy, w proporcji 70:30;
Zainicjuj zmienną $good\_answers = 0$ oraz słownik *statistics* z kluczami od 0 do 9, z wartościami będącymi podsłownikiem {'all':0, good:'0'}.
foreach *próbka w zbiorze testowym* **do**
    zapisz cyfrę próbki w zmiennej digit;
    zinkrementuj wartość 'all' w słowniku statystyk dla tej cyfry;
    na podstawie algorytmu KNN wygeneruj odpowiedź do zmiennej *answer*;
    if $answer = digit$ **then**
        zinkrementuj *good_answers*;
        zinkrementuj wartość *good* w słownik *statistics* dla cyfry *digit*;
    end
    else
        Wydrukuj informację o złej predykcji i o tym, którą cyfrę pomylono z którą.;
    end
end
Wydrukuj statystkę poprawnych odpowiedzi i wykres dla każdej cyfry.

Algorithm 1: Skrypt rozpoznający cyfry

Implementacja

Struktura plików w programie.



Poniżej implementacja metryki Minkowskiego, z której korzystam do obliczenia odległości między dwoma wektorami v1 i v2. W moim przypadku jeden wektor to aktualnie sprawdzana

cyfra ze zbioru testowego, a drugi to kolejna próbka ze zbioru walidacyjnego. Odległości liczone są na podstawie wszystkich pól rekordu oprócz ostatniego (ostatnie pole identyfikuje cyfrę).

```
    @staticmethod
    def minkowskiMetric(v1, v2, m):
      distance = 0
      for i in range(len(v1)-1):
        distance += abs(v1.iloc[i] - v2.iloc[i])**m
      distance = distance**(1/m)
      return distance
```

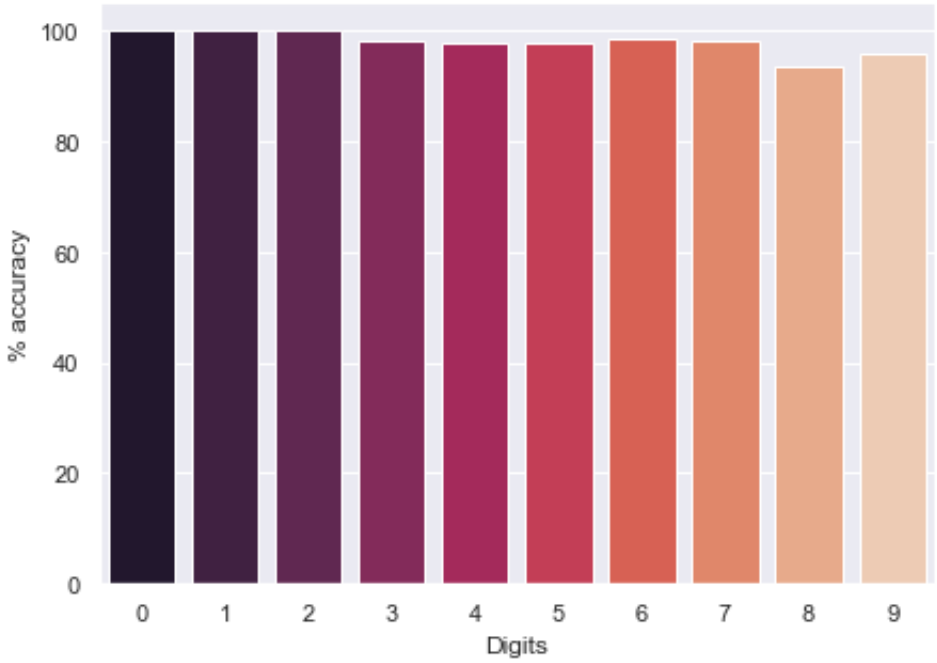
Niżej implementacja klasteryzacji. Widać wyszczególnione wszystkie cztery kroki algorytmu k-najbliższych sąsiadów, tj.

- Obliczanie odległości między próbkami
- Sortowanie odległości rosnąco
- Przeprowadzanie głosowania
- Zwrócenie wyniku

```
    def clustering(sample, X, k, m=2):
      distances = []
      classes = {i:0 for i in range(10)}
      # obliczanie odleglosci miedzy probkami
      for record in X.iloc:
        distances.append([KNN.minkowskiMetric(sample, record, m), record
            .Digit])

      # sortowanie
      distances = sorted(distances)
      # glosowanie
      for i in range(k):
        classes[distances[i][1]]+=1
      # zwrocenie wyniku
      return max(classes, key=classes.get)
```

Testy

```
TESTING WITH k=3 and m=1

Wrong prediction. Digit: 8, guessed: 3
Wrong prediction. Digit: 5, guessed: 9
Wrong prediction. Digit: 7, guessed: 9
Wrong prediction. Digit: 8, guessed: 3
Wrong prediction. Digit: 9, guessed: 8
Wrong prediction. Digit: 3, guessed: 7
Wrong prediction. Digit: 6, guessed: 1
Wrong prediction. Digit: 9, guessed: 8
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 4, guessed: 9
Accuracy: 98.14814814814815%
```



Najlepsze wyniki otrzymałem dla parametrów $k = 3$ i $m = 1$. Po losowym przetasowaniu bazy otrzymałem dokładność na poziomie 98%. Jak widać mylone są czasem cyfry o dość podobnym kształcie, np. 4 i 9 czy 9 i 8. Zauważyłem, że dla tak dobranych parametrów błędy rozkładają sięnieco bardziej równomiernie niż dla innych.

Eksperymenty

Przed wybraniem dla parametrów wartości 3 i 1 testowałem algorytm dla różnych innych wartości, dla k wartości od 1 do 5, dla m od 1 do 3, na sporo pomniejszonej bazie (po 30 rekordów do każdej cyfry) i otrzymałem poniższe wyniki.

Próba 1 27.04 21.02.txt - Notepad

```
TESTING WITH k=1 and m=1

Wrong prediction. Digit: 6, guessed: 1
Wrong prediction. Digit: 9, guessed: 1
Accuracy: 96.66666666666667%
TESTING WITH k=1 and m=2

Wrong prediction. Digit: 6, guessed: 1
Wrong prediction. Digit: 5, guessed: 6
Wrong prediction. Digit: 4, guessed: 9
Wrong prediction. Digit: 9, guessed: 1
Accuracy: 93.33333333333333%
TESTING WITH k=1 and m=3

Wrong prediction. Digit: 3, guessed: 9
Wrong prediction. Digit: 6, guessed: 1
Wrong prediction. Digit: 5, guessed: 6
Wrong prediction. Digit: 4, guessed: 8
Wrong prediction. Digit: 2, guessed: 3
Wrong prediction. Digit: 9, guessed: 1
Accuracy: 90.0%
TESTING WITH k=2 and m=1

Wrong prediction. Digit: 6, guessed: 1
Wrong prediction. Digit: 7, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 9, guessed: 7
Wrong prediction. Digit: 9, guessed: 1
Accuracy: 91.66666666666667%
TESTING WITH k=2 and m=2

Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 2, guessed: 0
Wrong prediction. Digit: 6, guessed: 1
Wrong prediction. Digit: 7, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 9, guessed: 7
Wrong prediction. Digit: 7, guessed: 2
Wrong prediction. Digit: 9, guessed: 1
Accuracy: 86.66666666666667%
TESTING WITH k=2 and m=3

Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 2, guessed: 0
Wrong prediction. Digit: 6, guessed: 1
Wrong prediction. Digit: 7, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 9, guessed: 7
Wrong prediction. Digit: 8, guessed: 0
Wrong prediction. Digit: 4, guessed: 8
Wrong prediction. Digit: 7, guessed: 2
Wrong prediction. Digit: 9, guessed: 1
Accuracy: 83.33333333333333%
TESTING WITH k=3 and m=1

Wrong prediction. Digit: 2, guessed: 9
Wrong prediction. Digit: 6, guessed: 1
Wrong prediction. Digit: 9, guessed: 7
Wrong prediction. Digit: 4, guessed: 9
Wrong prediction. Digit: 2, guessed: 3
Wrong prediction. Digit: 9, guessed: 1
Accuracy: 90.0%
TESTING WITH k=3 and m=2

Wrong prediction. Digit: 2, guessed: 0
Wrong prediction. Digit: 6, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 9, guessed: 7
Wrong prediction. Digit: 4, guessed: 9
Wrong prediction. Digit: 7, guessed: 2
Wrong prediction. Digit: 9, guessed: 1
Accuracy: 88.33333333333333%
TESTING WITH k=3 and m=3

Wrong prediction. Digit: 2, guessed: 0
Wrong prediction. Digit: 6, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 9, guessed: 7
Wrong prediction. Digit: 4, guessed: 5
Wrong prediction. Digit: 7, guessed: 2
Wrong prediction. Digit: 9, guessed: 1
Accuracy: 88.33333333333333%
TESTING WITH k=4 and m=1

Wrong prediction. Digit: 3, guessed: 9
Wrong prediction. Digit: 2, guessed: 9
Wrong prediction. Digit: 6, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 9, guessed: 7
Wrong prediction. Digit: 2, guessed: 3
Wrong prediction. Digit: 9, guessed: 1
Accuracy: 88.33333333333333%
TESTING WITH k=4 and m=2

Wrong prediction. Digit: 3, guessed: 8
Wrong prediction. Digit: 2, guessed: 0
Wrong prediction. Digit: 6, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 9, guessed: 7
Wrong prediction. Digit: 4, guessed: 9
Wrong prediction. Digit: 7, guessed: 2
Wrong prediction. Digit: 2, guessed: 3
Wrong prediction. Digit: 9, guessed: 1
Accuracy: 85.0%
TESTING WITH k=4 and m=3
```

Próba 2 4.05 10.48.txt - Notepad

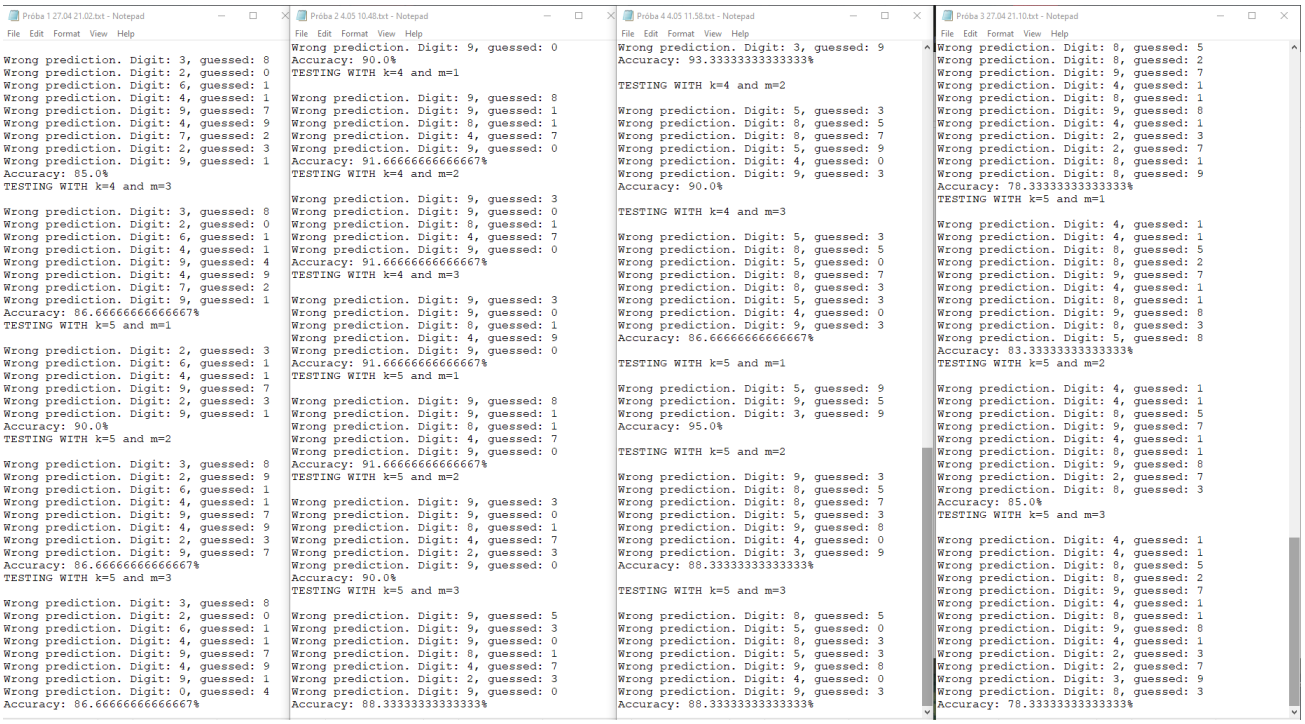
```
TESTING WITH k=1 and m=1

Wrong prediction. Digit: 9, guessed: 8
Wrong prediction. Digit: 9, guessed: 1
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 7, guessed: 9
Wrong prediction. Digit: 4, guessed: 7
Wrong prediction. Digit: 9, guessed: 0
Accuracy: 90.0%
TESTING WITH k=1 and m=2

Wrong prediction. Digit: 9, guessed: 3
Wrong prediction. Digit: 9, guessed: 0
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 7, guessed: 9
Wrong prediction. Digit: 4, guessed: 9
Wrong prediction. Digit: 9, guessed: 0
Accuracy: 90.0%
TESTING WITH k=1 and m=3

Wrong prediction. Digit: 9, guessed: 3
Wrong prediction. Digit: 9, guessed: 0
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 7, guessed: 9
Wrong prediction. Digit: 9, guessed: 0
Wrong prediction. Digit: 4, guessed: 9
Wrong prediction. Digit: 9, guessed: 0
Accuracy: 88.33333333333333%
TESTING WITH k=2 and m=1

Wrong prediction. Digit: 9, guessed: 3
Wrong prediction. Digit: 9, guessed: 1
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 4, guessed: 7
Wrong prediction. Digit: 9, guessed: 0
Accuracy: 91.66666666666667%
TESTING WITH k=2 and m=2

Wrong prediction. Digit: 9, guessed: 3
Wrong prediction. Digit: 9, guessed: 0
Wrong prediction. Digit: 9, guessed: 1
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 9, guessed: 0
Wrong prediction. Digit: 9, guessed: 5
Accuracy: 88.33333333333333%
TESTING WITH k=2 and m=3

Wrong prediction. Digit: 9, guessed: 3
Wrong prediction. Digit: 9, guessed: 1
Wrong prediction. Digit: 4, guessed: 9
Wrong prediction. Digit: 9, guessed: 0
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 7, guessed: 9
Wrong prediction. Digit: 4, guessed: 9
Wrong prediction. Digit: 9, guessed: 0
Accuracy: 90.0%
TESTING WITH k=3 and m=1

Wrong prediction. Digit: 9, guessed: 8
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 4, guessed: 7
Wrong prediction. Digit: 9, guessed: 0
Accuracy: 93.33333333333333%
TESTING WITH k=3 and m=2

Wrong prediction. Digit: 9, guessed: 3
Wrong prediction. Digit: 9, guessed: 0
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 4, guessed: 9
Wrong prediction. Digit: 9, guessed: 0
Wrong prediction. Digit: 9, guessed: 5
Accuracy: 90.0%
TESTING WITH k=3 and m=3

Wrong prediction. Digit: 9, guessed: 3
Wrong prediction. Digit: 9, guessed: 0
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 7, guessed: 9
Wrong prediction. Digit: 4, guessed: 9
Wrong prediction. Digit: 9, guessed: 0
Accuracy: 90.0%
TESTING WITH k=4 and m=1

Wrong prediction. Digit: 9, guessed: 8
Wrong prediction. Digit: 9, guessed: 1
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 4, guessed: 7
Wrong prediction. Digit: 9, guessed: 0
Accuracy: 91.66666666666667%
TESTING WITH k=4 and m=2

Wrong prediction. Digit: 9, guessed: 3
Wrong prediction. Digit: 9, guessed: 0
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 4, guessed: 7
Wrong prediction. Digit: 9, guessed: 0
Accuracy: 91.66666666666667%
TESTING WITH k=4 and m=3

Wrong prediction. Digit: 9, guessed: 3
Wrong prediction. Digit: 9, guessed: 0
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 4, guessed: 9
Wrong prediction. Digit: 9, guessed: 0
Accuracy: 91.66666666666667%
TESTING WITH k=5 and m=1

Wrong prediction. Digit: 9, guessed: 8
Wrong prediction. Digit: 9, guessed: 1
```

Próba 4 4.05 11.58.txt - Notepad

```
TESTING WITH k=1 and m=1

Wrong prediction. Digit: 5, guessed: 9
Wrong prediction. Digit: 3, guessed: 9
Accuracy: 96.66666666666667%

TESTING WITH k=1 and m=2

Wrong prediction. Digit: 5, guessed: 9
Wrong prediction. Digit: 4, guessed: 0
Accuracy: 96.66666666666667%

TESTING WITH k=1 and m=3

Wrong prediction. Digit: 8, guessed: 7
Wrong prediction. Digit: 1, guessed: 9
Wrong prediction. Digit: 5, guessed: 9
Wrong prediction. Digit: 4, guessed: 0
Accuracy: 93.33333333333333%

TESTING WITH k=2 and m=1

Wrong prediction. Digit: 3, guessed: 2
Wrong prediction. Digit: 4, guessed: 0
Wrong prediction. Digit: 5, guessed: 3
Wrong prediction. Digit: 9, guessed: 8
Wrong prediction. Digit: 9, guessed: 3
Accuracy: 91.66666666666667%

TESTING WITH k=2 and m=2

Wrong prediction. Digit: 3, guessed: 2
Wrong prediction. Digit: 4, guessed: 0
Wrong prediction. Digit: 5, guessed: 3
Wrong prediction. Digit: 4, guessed: 0
Wrong prediction. Digit: 9, guessed: 3
Accuracy: 91.66666666666667%

TESTING WITH k=2 and m=3

Wrong prediction. Digit: 3, guessed: 2
Wrong prediction. Digit: 8, guessed: 5
Wrong prediction. Digit: 5, guessed: 0
Wrong prediction. Digit: 4, guessed: 0
Wrong prediction. Digit: 4, guessed: 0
Wrong prediction. Digit: 8, guessed: 7
Wrong prediction. Digit: 8, guessed: 2
Wrong prediction. Digit: 8, guessed: 3
Wrong prediction. Digit: 1, guessed: 3
Wrong prediction. Digit: 5, guessed: 3
Wrong prediction. Digit: 4, guessed: 0
Wrong prediction. Digit: 9, guessed: 3
Accuracy: 80.0%
TESTING WITH k=3 and m=1

Wrong prediction. Digit: 5, guessed: 9
Wrong prediction. Digit: 3, guessed: 9
Accuracy: 96.66666666666667%

TESTING WITH k=3 and m=2

Wrong prediction. Digit: 5, guessed: 3
Wrong prediction. Digit: 4, guessed: 0
Wrong prediction. Digit: 3, guessed: 9
Accuracy: 95.0%

TESTING WITH k=3 and m=3

Wrong prediction. Digit: 5, guessed: 0
Wrong prediction. Digit: 5, guessed: 3
Wrong prediction. Digit: 4, guessed: 0
Wrong prediction. Digit: 9, guessed: 3
Wrong prediction. Digit: 3, guessed: 9
Accuracy: 91.66666666666667%

TESTING WITH k=4 and m=1

Wrong prediction. Digit: 3, guessed: 2
Wrong prediction. Digit: 5, guessed: 9
Wrong prediction. Digit: 9, guessed: 3
Wrong prediction. Digit: 3, guessed: 9
Accuracy: 93.33333333333333%

TESTING WITH k=4 and m=2

Wrong prediction. Digit: 5, guessed: 3
Wrong prediction. Digit: 8, guessed: 5
Wrong prediction. Digit: 8, guessed: 7
Wrong prediction. Digit: 5, guessed: 9
Wrong prediction. Digit: 4, guessed: 0
Wrong prediction. Digit: 9, guessed: 3
Accuracy: 90.0%

TESTING WITH k=4 and m=3

Wrong prediction. Digit: 5, guessed: 3
Wrong prediction. Digit: 8, guessed: 5
Wrong prediction. Digit: 5, guessed: 0
Wrong prediction. Digit: 8, guessed: 7
Wrong prediction. Digit: 8, guessed: 3
Wrong prediction. Digit: 5, guessed: 3
Wrong prediction. Digit: 4, guessed: 0
Wrong prediction. Digit: 9, guessed: 3
Accuracy: 86.66666666666667%

TESTING WITH k=5 and m=1
```

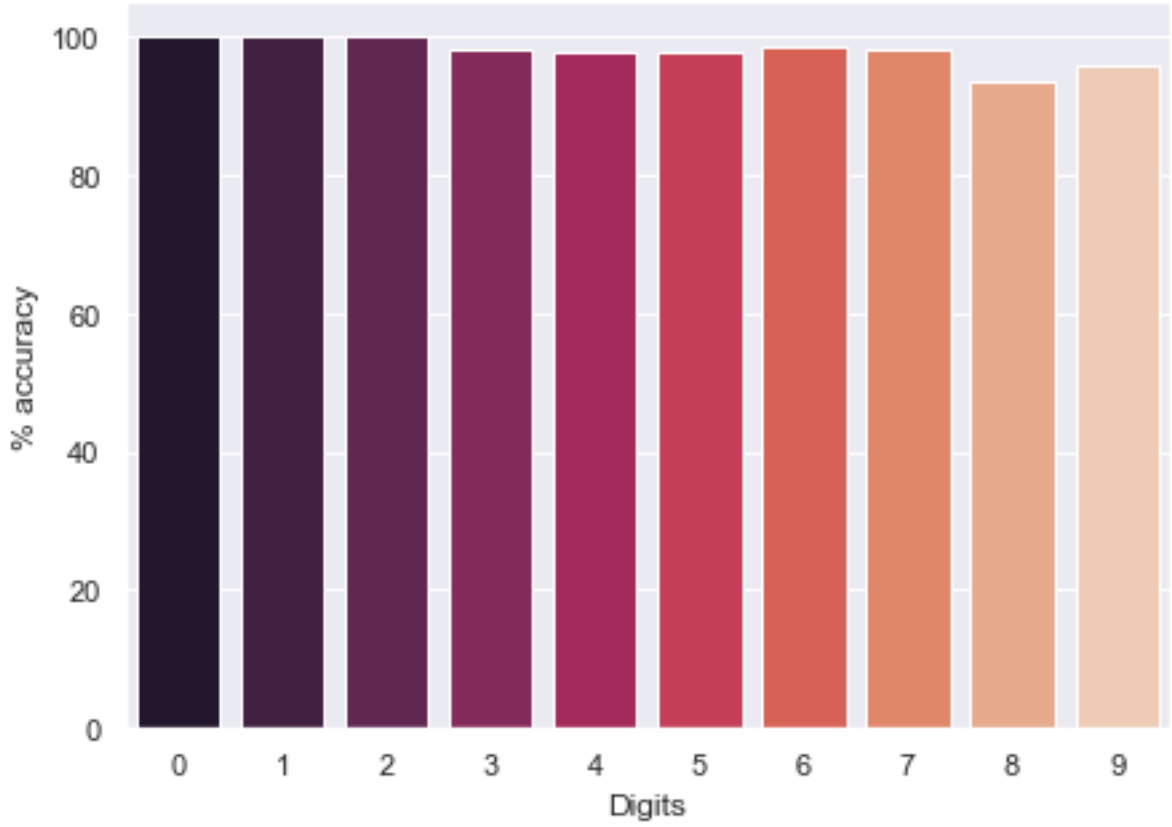
Próba 3 27.04 21.10.txt - Notepad

```
TESTING WITH k=1 and m=1

Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 3, guessed: 7
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 8, guessed: 9
Accuracy: 91.66666666666667%
TESTING WITH k=1 and m=2

Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 2
Wrong prediction. Digit: 9, guessed: 7
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 2, guessed: 7
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 8, guessed: 9
Accuracy: 85.0%
TESTING WITH k=1 and m=3

Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 2
Wrong prediction. Digit: 9, guessed: 5
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 9, guessed: 8
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 2, guessed: 7
Wrong prediction. Digit: 9, guessed: 0
Wrong prediction. Digit: 8, guessed: 5
Accuracy: 83.33333333333333%
TESTING WITH k=2 and m=1

Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 2
Wrong prediction. Digit: 9, guessed: 7
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 9, guessed: 8
Wrong prediction. Digit: 9, guessed: 3
Wrong prediction. Digit: 8, guessed: 3
Accuracy: 85.0%
TESTING WITH k=2 and m=2

Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 2
Wrong prediction. Digit: 9, guessed: 7
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 3
Wrong prediction. Digit: 5, guessed: 8
Accuracy: 86.66666666666667%
TESTING WITH k=3 and m=2

Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 2
Wrong prediction. Digit: 3, guessed: 2
Wrong prediction. Digit: 9, guessed: 5
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 3, guessed: 2
Wrong prediction. Digit: 2, guessed: 7
Wrong prediction. Digit: 8, guessed: 3
Wrong prediction. Digit: 5, guessed: 8
Accuracy: 80.0%
TESTING WITH k=3 and m=3

Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 2
Wrong prediction. Digit: 3, guessed: 2
Wrong prediction. Digit: 9, guessed: 4
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 9, guessed: 4
Wrong prediction. Digit: 2, guessed: 3
Wrong prediction. Digit: 2, guessed: 7
Wrong prediction. Digit: 8, guessed: 3
Wrong prediction. Digit: 5, guessed: 8
Accuracy: 78.33333333333333%
TESTING WITH k=4 and m=1

Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 5
Wrong prediction. Digit: 8, guessed: 2
Wrong prediction. Digit: 9, guessed: 5
Wrong prediction. Digit: 4, guessed: 1
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 9, guessed: 4
Wrong prediction. Digit: 2, guessed: 3
Wrong prediction. Digit: 2, guessed: 7
Wrong prediction. Digit: 8, guessed: 3
Wrong prediction. Digit: 5, guessed: 8
Accuracy: 83.33333333333333%
TESTING WITH k=5 and m=1
```

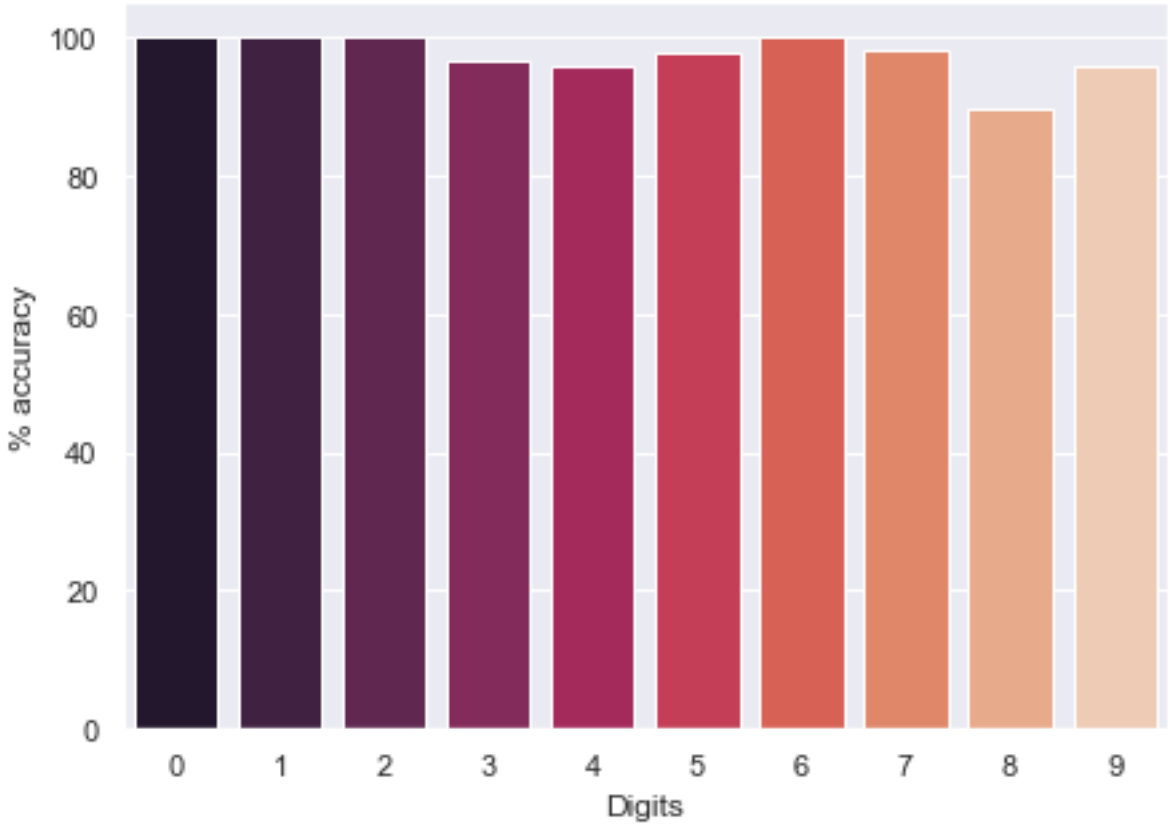


Na podstawie tych wyników wybrałem dwie najlepsze pary parametrów, dla których przeprowadziłem test na całej bazie, otrzymując następujące wyniki:

```
TESTING WITH k=3 and m=1

Wrong prediction. Digit: 8, guessed: 3
Wrong prediction. Digit: 5, guessed: 9
Wrong prediction. Digit: 7, guessed: 9
Wrong prediction. Digit: 8, guessed: 3
Wrong prediction. Digit: 9, guessed: 8
Wrong prediction. Digit: 3, guessed: 7
Wrong prediction. Digit: 6, guessed: 1
Wrong prediction. Digit: 9, guessed: 8
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 4, guessed: 9
Accuracy: 98.14814814814815%
```



```
TESTING WITH k=1 and m=1

Wrong prediction. Digit: 8, guessed: 3
Wrong prediction. Digit: 4, guessed: 9
Wrong prediction. Digit: 5, guessed: 3
Wrong prediction. Digit: 7, guessed: 9
Wrong prediction. Digit: 8, guessed: 3
Wrong prediction. Digit: 9, guessed: 8
Wrong prediction. Digit: 3, guessed: 7
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 3, guessed: 8
Wrong prediction. Digit: 9, guessed: 8
Wrong prediction. Digit: 8, guessed: 1
Wrong prediction. Digit: 4, guessed: 9
Wrong prediction. Digit: 8, guessed: 1
Accuracy: 97.5925925925926%
```



Stąd uznałem, że najlepsze wyniki dają parametry $k = 3$ i $m = 1$.

Wybranie wartości 3 dla parametru k sprawia, że głosowanie jest nieco bardziej demokratyczne, niż jeśli byłyby to mniejsze wartości. Warto zauważyć, że 3 to liczba nieparzysta,

co niweluje problem gdy algorytm rozważa dwie potencjalne cyfry: zostanie wybrana ta z dwoma głosami, a nie z jednym.

Dla parametru m wybór wartości 1 okazał się lepszym dla wskazania większych różnic między próbkami, co pozwoliło na zwiększenie dokładności algorytmu, mylił się rzadziej.

Pełen kod aplikacji

```
1     import pandas as pd
2     import numpy as np
3     import random
4     import seaborn as sns
5     import matplotlib.pyplot as plt
6
7     digits = pd.read_csv("digit_matrices.csv")
8     # przetasowanie
9     digits = digits.sample(frac=1).reset_index(drop=True)
10    g1=sns.violinplot(y="36",x="Digit",data=digits, inner='stick')
11    g1=sns.violinplot(y="20",x="Digit",data=digits, inner='stick')
12    g1=sns.violinplot(y="60",x="Digit",data=digits, inner='stick')
13    g1=sns.violinplot(y="19",x="Digit",data=digits, inner='stick')
14    g1=sns.violinplot(y="55",x="Digit",data=digits, inner='stick')
15
16    class DataProcessing:
17      @staticmethod
18      def shuffle(X):
19        return X.sample(frac=1).reset_index(drop=True)
20
21      #podzial zbioru
22      @staticmethod
23      def splitSet(X):
24        split = int(len(X)*0.7)
25        training_set = X[:split]
26        validation_set =X[split:]
27        return [training_set, validation_set]
28
29
30    class KNN:
31      @staticmethod
32      def minkowskiMetric(v1, v2, m):
33        distance = 0
34        for i in range(len(v1)-1):
35          distance += abs(v1.iloc[i] - v2.iloc[i])**m
36        distance = distance**(1/m)
37        return distance
38
39        def clustering(sample, X, k, m=2):
40        distances = []
41        classes = {i:0 for i in range(10)}
42        # obliczanie odleglosci miedzy probkami
43        for record in X.iloc:
44          distances.append([KNN.minkowskiMetric(sample, record, m),
              record.Digit])
45
46        # sortowanie
47        distances = sorted(distances)
48        # glosowanie
49        for i in range(k):
50          classes[distances[i][1]]+=1
51        # zwrocenie wyniku
```

```
52          return max(classes, key=classes.get)
53
54      def test(training_set, validation_set, k=4, m=2):
55
56        good_answers = 0
57        statistics = {i: {'all': 0, 'good': 0} for i in range(10)}
58        for sample in validation_set.iloc:
59          digit = sample.Digit
60          statistics[digit]['all'] += 1
61          answer = KNN.clustering(sample, training_set, k, m)
62          if answer == digit:
63            good_answers += 1
64            statistics[digit]['good'] += 1
65          else:
66            print(f"Wrong prediction. Digit: {digit}, guessed: {answer}")
67        print(f"Accuracy: {100*good_answers/len(validation_set)}%")
68        print_stats(statistics)
69
70      def print_stats(statistics):
71        x = [i for i in statistics.keys()]
72        y = []
73        for i in x:
74          y.append(100*statistics[i]['good'] / statistics[i]['all'])
75
76        sns.set_theme()
77        f, ax1 = plt.subplots(1,1, figsize=(7,5), sharex=True)
78
79        fig = sns.barplot(x=x, y=y, palette='rocket', ax=ax1)
80        fig.set(xlabel='Digits', ylabel='% accuracy')
81        plt.show()
82
83      k=3
84      m=1
85      print(f"TESTING WITH k={k} and m={m}\n")
86          test(training_set, validation_set, k, m)
```