Sprawozdanie - elementy inteligencji obliczeniowej (EIO)	
Prowadzący: *mgr inż. Anna Labijak-Kowalska*	Grupa: ***L9***
Temat Ćwiczeń: *Klasyfikator **K-Means***	
Autorzy: *Daniel Zdancewicz [145317]*	

1 Wstęp - Cel Ćwiczenia

Celem ćwiczenia była implementacja klasyfikatora K-Means.

2 Implementacja algorytmu k-means

```
def k_means(points, /, *, n_clusters: int, iterations: int, tolerance: float):
  def distance(x, y):
    return np.sqrt(np.sum((x - y) ** 2, axis=1))

  centroids = np.random.uniform(
    np.min(points, axis=0),
    np.max(points, axis=0),
    size=(n_clusters, points.shape[1])
  )

  labels = []
  previous = centroids
  for i in range(iterations):
    labels = np.argmin([
      distance(centroid, points) for centroid in centroids
    ], axis=0)

    centroids = np.array([
      np.mean(points[labels == label], axis=0)
      if np.sum(labels == label)
      else centroids[label]
      for label in range(n_clusters)
    ])

    if np.all(np.abs(previous - centroids) < tolerance):
      print(f"k-means algorithm early stopped at: {i}")
      break
```

```
    previous = centroids
  return (labels, centroids)
```

2.1 Odczyt i preprocessing

Odczyt odbywa się przy wykorzystaniu metody reader z biblioteki csv.

Normalizacja odbyła się poprzez wykorzystanie metody fit_transform dostępnej w sklearn.preprocessing.StandardScaler wykonany na wartościach numerycznych.

```
def preprocess(points):
  return skp.StandardScaler().fit_transform(points)

def read_file(path: str):
  with open(path) as file:
    return preprocess(tuple(csv.reader(file)))
```

2.2 Inicjalizaja - przypadkowe centroidy

Przy rozpoczęciu algorytmu są wybierane przypadkowe punkty jako początkowe wybrane środki centroidów.

```
def k_means(points, /, *, n_clusters: int, iterations: int, tolerance: float):
  ...
  centroids = np.random.uniform(
    np.min(points, axis=0),
    np.max(points, axis=0),
    size=(n_clusters, points.shape[1])
  )
  ...
```

2.3 Kryterium zatrzymania - Maksymalna liczba iteracji

Kryterium zatrzymania zostało zaimplementowane poprzez zatrzymanie iteracji treningowych po osiągnięciu pewnej liczbie naturalnej podanej przez parametr iterations.

```
def k_means(points, /, *, n_clusters: int, iterations: int, tolerance: float):
  ...
  # Kryterium zatrzymania.
  for i in range(iterations):
    ...
  ...
```

2.4 Kryterium zatrzymania - Zbieżność centroidów

Kryterium zbieżności jest zaimplementowane zatrzymanie pętli treningowej po osiągnieciu pewnej zbieżności określonej pewną liczbą rzeczywistą podanej przez parametr tolerance.

```python
def k_means(points, /, *, n_clusters: int, iterations: int, tolerance: float):
  ...

  previous = centroids
  for i in range(iterations):
    ...
    # Kryterium zatrzymania - Pokrycie.
    if np.all(np.abs(previous - centroids) < tolerance):
      print(f"k-means algorithm early stopped at: {i}")
      break
    previous = centroids
  ...
```

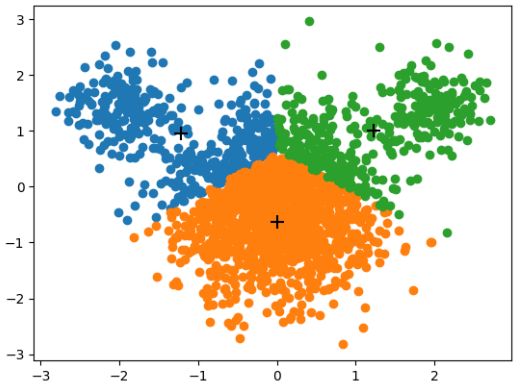
3 Użycie algorytmu na zbiorze mouse.csv oraz cereal.csv

3.1 Weryfikacja poprawności - mouse.csv

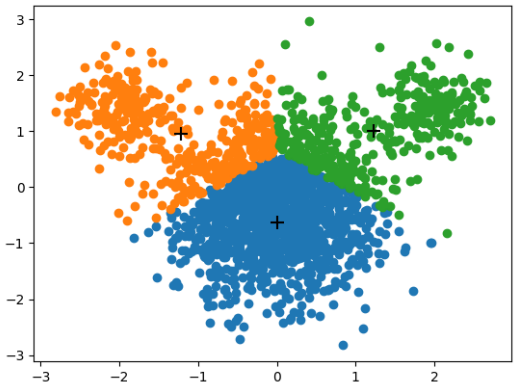
Zbiór danych mouse.csv dostarczony wcześniej przez prowadzącego opisuje punkty poprzez 2 kolumny wskazujące koordynaty x i y.

- Pierwsza kolumna – Koordynat X punktu.
- Druga kolumna – Koordynat X punktu.

Poprawność została przetestowana poprzez zbliżenia uzyskanych grup przez klasyfikator z biblioteki sklearn oraz przez klasyfikator z implementacji własnej.



Rysunek 1: Centroidy uzyskane przez scikit dla n_clusters = 3



Rysunek 2: Centroidy uzyskane przez implementację własną dla n_clusters = 3

Jak widać na podanych obrazach widać ten sam rozkład centroidów co może oznaczać poprawność algorytmu.

3.2 Tworzenie grup na bazie cerreal.csv

Zbiór danych cereal.csv dostarczony wcześniej przez prowadzącego opisuje 16 parametrów, z których opuszczone zostały kolumny name oraz mfr. te były znormalizowane przez MinMaxScaler dostępny przez sklearn.preprocessing.

Grupowanie odbywa się przez

```
metaparams = {"n_clusters": 3, "iterations": 100, "tolerance": 1e-6}
(labels, centroids) = k_means(normalized, **metaparams)

counts = np.asarray(np.unique(labels, return_counts=True)).T
print(f"""
Number of cereals within each cluster:
 {counts}

Labels:
 {labels}

Centroids:
""")
for (centroid, descriptor) in enumerate(map(lambda centroid: zip(normalized.columns, cen
  print(f"- Centroid No. '{centroid}'")
  for (column, value) in descriptor:
    print(f"  - {column: <10}: {value*100:.2f}%")
  print()
```

3.3 Przykładowe standardowe wyjście

```
k-means algorithm early stopped at: 9

Number of cereals within each cluster:
 [[ 0 36]
 [ 1 20]
 [ 2 21]]

Labels:
 [0 0 0 0 0 1 1 0 2 0 1 2 1 0 1 2 2 1 1 0 2 0 0 0 1 1 2 0 0 1 1 1 0 0 0 1 1
 1 0 0 2 0 1 2 0 0 0 2 1 0 0 0 0 0 2 2 0 2 0 0 0 2 2 2 2 2 1 2 2 0 0 0 0 1
 2 2 1]

Centroids:

- Centroid No. '0'
  - type      : 0.00%
  - calories  : 55.56%
  - protein   : 39.44%
```

  - fat : 26.67%
  - sodium : 52.69%
  - fiber : 23.61%
  - carbo : 64.93%
  - sugars : 50.17%
  - potass : 41.79%
  - vitamins : 36.81%
  - shelf : 97.22%
  - weight : 59.58%
  - cups : 36.80%
  - rating : 34.48%

- Centroid No. '1'
  - type : 0.00%
  - calories : 55.91%
  - protein : 10.00%
  - fat : 20.00%
  - sodium : 53.12%
  - fiber : 3.57%
  - carbo : 56.25%
  - sugars : 78.44%
  - potass : 13.52%
  - vitamins : 25.00%
  - shelf : 35.00%
  - weight : 50.00%
  - cups : 50.52%
  - rating : 13.54%

- Centroid No. '2'
  - type : 14.29%
  - calories : 41.13%
  - protein : 36.19%
  - fat : 9.52%
  - sodium : 42.04%
  - fiber : 12.48%
  - carbo : 73.41%
  - sugars : 20.83%
  - potass : 23.02%
  - vitamins : 16.67%
  - shelf : 21.43%
  - weight : 44.43%
  - cups : 56.30%
  - rating : 47.32%

3.4 Wnioski

Jak widać centroidy tworzą skupienia wokół kilku atrybutów. Wyróżnione zostały te, które przekroczyły wagę 0.5 (50%).

- Formują się 3 centroidy: Duży(36), Średni(20) i Średni (21).
- Pierwszy - Duży(36) – klasyfikacja płatków ze skupieniem na wagę, węglowodany, i na której półce się znajdowały:
  - calories – 55.56%.
  - sodium – 52.69%.
  - carbo – 64.93%.
  - sugars – 50.17%.
  - shelf – 97.22%.
  - weight – 59.58%.
- Drugi - Średni(20) – klasyfikacja płatów z dużą zawartością cukrów:
  - calories – 55.91%.
  - sodium – 53.12%.
  - carbo – 56.25%.
  - sugars – 78.44%.
  - cups – 50.52%.
- Trzeci - Średni(21) – klasyfikacja płatków z dużą zawartością węglowodanów:
  - carbo – 73.41%.
  - cups – 56.30%.

Jak widać wspólne, ważne są kalorie, węglowodany oraz cups czyli wielkość jednej porcji.