Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного автономного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ, информационные технологи»

ЛАБОРАТОРНАЯ РАБОТА

«Вычисление отступов по выборке»

по дисциплине: «Методы машинного обучения»

Выполнил: студент группы ИУК4-72Б _____ Губин Е.В.
(Подпись) (Фамилия И.О.)

Проверил(-а): _____ Семененко М.Г.
(Подпись) (Фамилия, И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2025

Цель: формирование навыков использования алгоритма k-ближайших соседей, STOLP и вычислить отступы для точек.

Задачи:

1. Применить алгоритм k-ближайших соседей к объекту выборки.
2. Применить алгоритм STOLP для отбора эталонных объектов.
3. Вычислить отступы для точек.

Задание:

1. Сгенерировать случайную выборку из 10 объектов с классами +1 и -1 и одним количественным признаком.

2. Выбрать один из объектов в сгенерированной выборке и рассчитать для него все отступы для алгоритма 2NN.

3. Применить алгоритм STOLP для отбора эталонных объектов.

4. Вычислить отступы для конкретной точки.

Ход выполнения работы:

Задание 1:

Составим скрипт на python-скрипт для генерации выборки.

main.py

```
import numpy as np
import matplotlib.pyplot as plt

# Генерация данных
np.random.seed(10)
X = np.random.uniform(0, 10, 10)
y = np.random.choice([-1, 1], 10)
sort_indices = np.argsort(X)
X = X[sort_indices]
y = y[sort_indices]
print("Данные:")
for i, (feature, label) in enumerate(zip(X, y)):
    print(f"{i}: {feature:.2f}, {label}")

plt.figure(figsize=(12, 3))
colors = ['red' if label == -1 else 'blue' for label in y]
plt.scatter(X, [0]*10, c=colors, s=100, alpha=0.7)
plt.yticks([])
plt.xlabel('X')
plt.title('Метки -1 (red), 1 (blue)')
for i, (x, label) in enumerate(zip(X, y)):
    plt.text(x, 0.02, f'{i}', ha='center', va='bottom')
plt.grid(True, alpha=0.3)
plt.show()
```

Результат выполнения скрипта:

```
Данные:
0: 0.21, 1
1: 0.88, -1
```

```
2: 1.69, 1
3: 1.98, -1
4: 2.25, 1
5: 4.99, 1
6: 6.34, -1
7: 7.49, -1
8: 7.61, -1
9: 7.71, -1
```

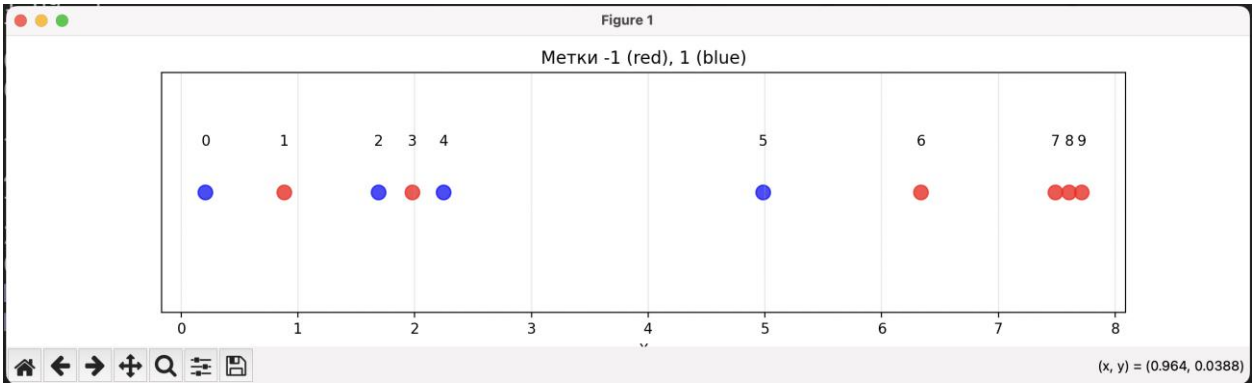

Рис. 1 «Результат выполнения скрипта первого задания»

Задание 2:

Составим скрипт на python-скрипт для выбора одного из объектов в сгенерированной выборке и расчёта для него отступов алгоритма 2NN.

main.py

```
# 2NN-margin
def calculate_2nn_margin(X, y, object_index):
    target_x = X[object_index]
    target_y = y[object_index]
    distances = []
    for i, (x, label) in enumerate(zip(X, y)):
        if i != object_index:
            dist = abs(x - target_x)
            distances.append((dist, label, i))
    distances.sort(key=lambda x: x[0])
    neighbor1_dist, neighbor1_label, neighbor1_idx = distances[0]
    neighbor2_dist, neighbor2_label, neighbor2_idx = distances[1]
    print(f"{object_index} ({target_x:.2f}, {target_y}):")
    print(f"  1NN: {neighbor1_idx}, {neighbor1_dist:.2f}, {neighbor1_label}")
    print(f"  2NN: {neighbor2_idx}, {neighbor2_dist:.2f}, {neighbor2_label}")
    same_class_distances = [dist for dist, label, idx in distances[:2] if label
== target_y]
    diff_class_distances = [dist for dist, label, idx in distances[:2] if label
!= target_y]
    if same_class_distances and diff_class_distances:
        margin = min(diff_class_distances) - min(same_class_distances)
    elif same_class_distances:
        margin = float('inf')
    else:
        margin = -float('inf')
    print(f"  2NN margin: {margin:.2f}")
    if margin > 0:
        print("  object OK")
    elif margin < 0:
        print("  object ERROR")
```

```
    else:
        print("  object on margin")
    return margin

selected_object = 4
margin2nn = calculate_2nn_margin(X, y, selected_object)

print("=" * 50)
print("Margins for all objects:")
print("=" * 50)
margins = []
for i in range(len(X)):
    margin = calculate_2nn_margin(X, y, i)
    margins.append(margin)
```

Результат выполнения скрипта:

```
4 (2.25, 1):
  1NN: 3, 0.27, -1
  2NN: 2, 0.56, 1
  2NN margin: -0.29
  object ERROR
==================================================
Margins for all objects:
==================================================
0 (0.21, 1):
  1NN: 1, 0.68, -1
  2NN: 2, 1.48, 1
  2NN margin: -0.81
  object ERROR
1 (0.88, -1):
  1NN: 0, 0.68, 1
  2NN: 2, 0.81, 1
  2NN margin: -inf
  object ERROR
2 (1.69, 1):
  1NN: 3, 0.29, -1
  2NN: 4, 0.56, 1
  2NN margin: -0.27
  object ERROR
3 (1.98, -1):
  1NN: 4, 0.27, 1
  2NN: 2, 0.29, 1
  2NN margin: -inf
  object ERROR
4 (2.25, 1):
  1NN: 3, 0.27, -1
  2NN: 2, 0.56, 1
  2NN margin: -0.29
  object ERROR
5 (4.99, 1):
  1NN: 6, 1.35, -1
  2NN: 7, 2.50, -1
  2NN margin: -inf
  object ERROR
6 (6.34, -1):
  1NN: 7, 1.15, -1
  2NN: 8, 1.27, -1
  2NN margin: inf
  object OK
7 (7.49, -1):
  1NN: 8, 0.12, -1
  2NN: 9, 0.23, -1
```

```
  2NN margin: inf
  object OK
8 (7.61, -1):
  1NN: 9, 0.11, -1
  2NN: 7, 0.12, -1
  2NN margin: inf
  object OK
9 (7.71, -1):
  1NN: 8, 0.11, -1
  2NN: 7, 0.23, -1
  2NN margin: inf
  object OK
```

Задание 3:

Составим скрипт на python-скрипт, реализующий алгоритм STOLP, для отбора эталонных объектов.

main.py

```
      # STOLP
      def    stolp_algorithm(X,    y,    theta_plus=0.1,    theta_minus=-0.1,
max_iter=100):
          n = len(X)
          # Находим маргины
          margins = []
          for i in range(n):
              margin = calculate_2nn_margin(X, y, i)
              margins.append(margin)
          prototypes = []
          prototype_indices = []

          # Выбор опорных объектов разных классов с максимальным margin
          plus_margins = [(margin, i) for i, margin in enumerate(margins) if
y[i] == 1]
          if plus_margins:
              best_plus_margin, best_plus_idx = max(plus_margins)
              prototypes.append((X[best_plus_idx], y[best_plus_idx]))
              prototype_indices.append(best_plus_idx)
              print(f"+1: {best_plus_idx}")
          minus_margins = [(margin, i) for i, margin in enumerate(margins) if
y[i] == -1]
          if minus_margins:
              best_minus_margin, best_minus_idx = max(minus_margins)
              prototypes.append((X[best_minus_idx], y[best_minus_idx]))
              prototype_indices.append(best_minus_idx)
              print(f"-1: {best_minus_idx}")

          for iteration in range(max_iter):
              current_margins = []
              for i in range(n):
                  if i in prototype_indices:
                      continue
                  # margin относительно прототипов
                  distances_to_prototypes = []
                  for j, (proto_x, proto_y) in enumerate(prototypes):
                      dist = abs(X[i] - proto_x)
                      distances_to_prototypes.append((dist, proto_y, j))
                  distances_to_prototypes.sort(key=lambda x: x[0])
                  nearest_prototypes = distances_to_prototypes[:2]
```

```
                same_class_dists = [dist for dist, label, idx in
nearest_prototypes if label == y[i]]
                diff_class_dists = [dist for dist, label, idx in
nearest_prototypes if label != y[i]]
                if same_class_dists and diff_class_dists:
                    margin = min(diff_class_dists) - min(same_class_dists)
                elif same_class_dists:
                    margin = float('inf')
                else:
                    margin = -float('inf')
                current_margins.append((margin, i))
            if not current_margins:
                break
            errors = [(margin, idx) for margin, idx in current_margins if
margin < theta_minus]
            if not errors:
                break
            worst_margin, worst_idx = min(errors, key=lambda x: x[0])
            print(f"Добавляется в прототипы: {worst_idx}, {y[worst_idx]}")
            prototypes.append((X[worst_idx], y[worst_idx]))
            prototype_indices.append(worst_idx)
            # возможно, еще позитивные
            good_objects = [(margin, idx) for margin, idx in current_margins
if margin > theta_plus and idx not in prototype_indices]
            if good_objects and len(prototypes) < n:
                best_margin, best_idx = max(good_objects)
                prototypes.append((X[best_idx], y[best_idx]))
                prototype_indices.append(best_idx)
                print(f"Добавляется в прототипы: {best_idx} (good margin)")
        return prototype_indices, prototypes

    print("=" * 60)
    print("STOLP алгоритм")
    print("=" * 60)
    prototype_indices, prototypes = stolp_algorithm(X, y)

    print("=" * 50)
    print("STOLP прототипы")
    print("=" * 50)
    print(f"N:      {len(X)},      Прототипы:      {len(prototype_indices)},
({len(prototype_indices)/len(X):.1%})")
    for i, idx in enumerate(prototype_indices):
        print(f"{i}: {idx}, {X[idx]:.2f}, {y[idx]}")

    plt.figure(figsize=(12, 4))
    plt.subplot(1, 2, 1)
    colors = ['red' if label == -1 else 'blue' for label in y]
    plt.scatter(X, [0]*10, c=colors, s=100, alpha=0.7)
    plt.yticks([])
    plt.xlabel('X')
    plt.title('До STOLP')
    for i, (x, label) in enumerate(zip(X, y)):
        plt.text(x, 0.02, f'{i}', ha='center', va='bottom')
    plt.grid(True, alpha=0.3)

    plt.subplot(1, 2, 2)
    all_colors = ['red' if label == -1 else 'blue' for label in y]
    prototype_colors = ['darkred' if label == -1 else 'darkblue' for _, label
in prototypes]
    plt.scatter(X, [0]*10, c=all_colors, s=100, alpha=0.2)
    prototype_X = [X[i] for i in prototype_indices]
    prototype_y = [y[i] for i in prototype_indices]
    plt.scatter(prototype_X,    [0]*len(prototype_X),    c=prototype_colors,
s=100)
```

```
      plt.yticks([])
      plt.xlabel('X')
      plt.title('После STOLP')
      for i, (x, label) in enumerate(zip(X, y)):
          if i in prototype_indices:
              plt.text(x,    0.02,    f'{i}',    ha='center',    va='bottom',
fontweight='bold')
          else:
              plt.text(x, 0.02, f'{i}', ha='center', va='bottom', alpha=0.5)
      plt.grid(True, alpha=0.3)
      plt.tight_layout()
      plt.show()
```

Результат выполнения скрипта:

```
============================================================
STOLP алгоритм
============================================================
0 (0.21, 1):
  1NN: 1, 0.68, -1
  2NN: 2, 1.48, 1
  2NN margin: -0.81
  object ERROR
1 (0.88, -1):
  1NN: 0, 0.68, 1
  2NN: 2, 0.81, 1
  2NN margin: -inf
  object ERROR
2 (1.69, 1):
  1NN: 3, 0.29, -1
  2NN: 4, 0.56, 1
  2NN margin: -0.27
  object ERROR
3 (1.98, -1):
  1NN: 4, 0.27, 1
  2NN: 2, 0.29, 1
  2NN margin: -inf
  object ERROR
4 (2.25, 1):
  1NN: 3, 0.27, -1
  2NN: 2, 0.56, 1
  2NN margin: -0.29
  object ERROR
5 (4.99, 1):
  1NN: 6, 1.35, -1
  2NN: 7, 2.50, -1
  2NN margin: -inf
  object ERROR
6 (6.34, -1):
  1NN: 7, 1.15, -1
  2NN: 8, 1.27, -1
  2NN margin: inf
  object OK
7 (7.49, -1):
  1NN: 8, 0.12, -1
  2NN: 9, 0.23, -1
  2NN margin: inf
  object OK
8 (7.61, -1):
  1NN: 9, 0.11, -1
  2NN: 7, 0.12, -1
  2NN margin: inf
  object OK
```

```
9 (7.71, -1):
  1NN: 8, 0.11, -1
  2NN: 7, 0.23, -1
  2NN margin: inf
  object OK
+1: 2
-1: 9
Добавляется в прототипы: 1, -1
Добавляется в прототипы: 0 (good margin)
Добавляется в прототипы: 3, -1
Добавляется в прототипы: 8 (good margin)
Добавляется в прототипы: 5, 1
Добавляется в прототипы: 7 (good margin)
Добавляется в прототипы: 4, 1
Добавляется в прототипы: 6 (good margin)
================================================
STOLP прототипы
================================================
N: 10, Прототипы: 10, (100.0%)
0: 2, 1.69, 1
1: 9, 7.71, -1
2: 1, 0.88, -1
3: 0, 0.21, 1
4: 3, 1.98, -1
5: 8, 7.61, -1
6: 5, 4.99, 1
7: 7, 7.49, -1
8: 4, 2.25, 1
9: 6, 6.34, -1
```

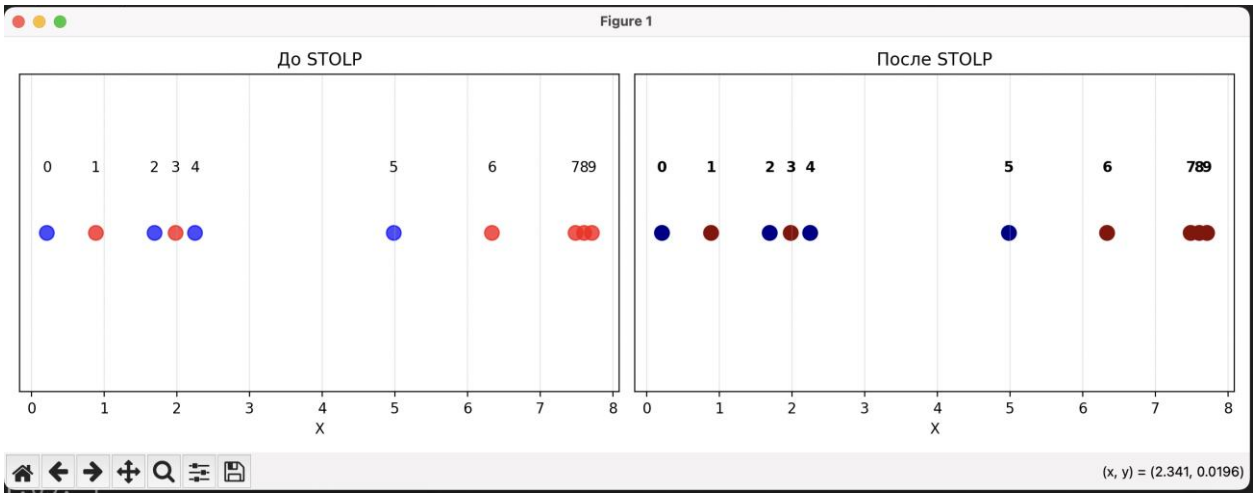

Рис. 2 «Результат выполнения скрипта третьего задания»

Таблица 1. Значения отступов для точки 4.

	0	**1**	**2**	**3**	**5**	**6**	**7**	**8**	**9**
4	3,89	2,602	1,712	0,778	0,999	1,951	3,355	4,222	5,276

На основе вычисления 2NN-отступов (margins) для всех объектов было выявлено:

- часть точек имеет отрицательные отступы (margin < 0) — они находятся рядом с объектами другого класса → классифицируются неправильно или лежат в зоне пересечения классов;
- несколько точек имеют положительный бесконечный отступ (margin = +∞) — они окружены соседями своего класса → являются типичными представителями класса, хорошими кандидатами в прототипы.

Алгоритм STOLP, используя значения отступов, выбрал:

1. по одному лучшему объекту в каждом классе — основные эталоны (опорные объекты);
2. добавил дополнительные прототипы только для тех точек, у которых margin < θ_- (сильные ошибки) — чтобы исправить ошибки классификации;

Вывод: в ходе выполнения лабораторной работы были сформированы практические навыки использования алгоритма k-ближайших соседей и STOLP.