Министерство цифрового развития

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

# Отчёт

по лабораторной работе № 2 «Метод k-ближайших соседей»

Выполнил:

студент группы ИП-312

Прозоренко К.В

Работу проверил: старший преподаватель кафедры ПМиК
Дементьева К.И.

Новосибирск 2025 г.

## Введение (задание)

Разработка классификатора на основе метода k ближайших соседей.

**Основная часть:**

**1.** Загружаем набор данных в соответствии с вариантом:

$N$ф = (($Nc$ +2) $mod$ 5) + 1, где $N$ф – номер файла, $Nc$ − номер по списку группы.

Мой номер в списке - 16.

$N$ф = ((16 + 2) mod 5) + 1 = 4

data4.csv

1 to 10 of 10000 entries Filter

| MrotInHour | Salary | Class |
|---|---|---|
| 4 | 13270 | 1 |
| 3 | 24786 | 1 |
| 9 | 3651 | 0 |
| 10 | 10622 | 0 |
| 3 | 15634 | 1 |
| 6 | 22725 | 1 |
| 5 | 16817 | 1 |
| 2 | 7933 | 1 |
| 2 | 4422 | 0 |
| 10 | 5985 | 0 |

Show 10 per page 1 2 10 100 900 990 1000

На основе этих данных необходимо обучить разработанный классификатор, отразив метод подбора параметров в соответствии с вариантом (k, i, q, h), и протестировать метод на тестовой выборке. Вариант алгоритма выбирается следующим образом:

$N$в =($Nc$ $mod$ 3)+1, где $N$в – номер варианта, $Nc$ − номер по списку группы

$N$в = (16 $mod$ 3) + 1 = 2

Метод парзеновского окна с фиксированным h

В данном варианте необходимо использовать функцию ядра K(z),

выбранную следующим образом:

$N$я = ($Nc * 6 + 13) \; mod \; 8 \; mod \; 3) + 1$

$N$я = ((16 $*$ 6 + 13) $mod$ 8 $mod$ 3) + 1 = 3

3. *П – прямоугольное* $K(x) = [r \leq 1]$

**ТАБЛИЦА РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ:**

```
======================================================================
ТАБЛИЦА РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ
======================================================================
 Разбиение  Размер обуч.  Размер тест. Accuracy Precision Recall F1-Score Время (с)
         1          7000          3000   0.9973    0.9917 0.9950   0.9934      1.38
         2          7000          3000   0.9980    0.9918 0.9983   0.9951      1.36
         3          7000          3000   0.9987    0.9967 0.9967   0.9967      1.21
         4          7000          3000   0.9980    0.9918 0.9983   0.9950      1.20
         5          7000          3000   0.9963    0.9853 0.9967   0.9909      1.20
         6          7000          3000   0.9993    1.0000 0.9967   0.9983      1.22
         7          7000          3000   0.9980    0.9967 0.9934   0.9950      1.19
         8          7000          3000   0.9990    0.9951 1.0000   0.9975      1.23
         9          7000          3000   0.9967    0.9885 0.9950   0.9918      1.19
        10          7000          3000   0.9983    0.9950 0.9967   0.9959      1.39

======================================================================
ИТОГОВАЯ СТАТИСТИКА
======================================================================
Средняя Accuracy:  0.9980 ± 0.0009
Средний F1-Score:  0.9950 ± 0.0022
Мин. Accuracy:     0.9963
Макс. Accuracy:    0.9993
```

**1. Метод парзеновского окна с прямоугольным ядром успешно реализован**

**2. Оптимальный параметр ширины окна: h = 0.1**

**3. Средняя точность на 10 разбиениях: 99.80%**

**4. Низкая вариативность результатов (std = 0.0009)**

**5. Классификатор стабилен и показывает высокую точность**

**6. Метод эффективно работает даже на несбалансированных данных**

***Ссылка на Google Collab:*** https://drive.google.com/file/d/1H5tKNIKr4aHpev9DOZuF-qXN9NaIIuJB/view?usp=sharing

# Код программы:

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import time
import warnings
warnings.filterwarnings('ignore')


df = pd.read_csv('data4.csv')
X = df[['MrotInHour', 'Salary']].values
y = df['Class'].values

print("=" * 70)
print("ШАГ 1: ЗАГРУЗКА И АНАЛИЗ ДАННЫХ")
print("=" * 70)
print(f"Размерность датасета: {df.shape}")
print(f"Признаки: {list(df.columns[:-1])}")
print(f"Целевая переменная: {df.columns[-1]}")
print(f"\nРаспределение классов:")
print(f"  Класс 0: {sum(y==0)} образцов ({sum(y==0)/len(y)*100:.2f}%)")
print(f"  Класс 1: {sum(y==1)} образцов ({sum(y==1)/len(y)*100:.2f}%)")
print(f"\nВЫВОД: Данные несбалансированы, класс 0 доминирует.\n")


class ParzenWindowClassifier:

    def __init__(self, h=1.0):
        self.h = h
        self.X_train = None
        self.y_train = None
        self.scaler = StandardScaler()

    def rectangular_kernel(self, r):
        return (r <= 1.0).astype(float)

    def fit(self, X_train, y_train):
        self.X_train = self.scaler.fit_transform(X_train)
        self.y_train = y_train
        self.classes = np.unique(y_train)
        return self

    def predict_single(self, x):
        x_scaled = self.scaler.transform(x.reshape(1, -1))[0]

        distances = np.sqrt(np.sum((self.X_train - x_scaled)**2, axis=1))

        normalized_distances = distances / self.h

        weights = self.rectangular_kernel(normalized_distances)
```

```
        class_weights = {}
        for c in self.classes:
            class_mask = (self.y_train == c)
            class_weights[c] = np.sum(weights[class_mask])

        if sum(class_weights.values()) == 0:
            return np.argmax(np.bincount(self.y_train))

        return max(class_weights, key=class_weights.get)

    def predict(self, X_test):
        return np.array([self.predict_single(x) for x in X_test])

    def score(self, X_test, y_test):
        y_pred = self.predict(X_test)
        return np.mean(y_pred == y_test)

print("=" * 70)
print("ШАГ 2: РЕАЛИЗАЦИЯ КЛАССИФИКАТОРА")
print("=" * 70)
print("✓ Реализован ParzenWindowClassifier с прямоугольным ядром")
print("✓ Используется нормализация признаков (StandardScaler)")
print("✓ Метод классификации: взвешенное голосование\n")

print("=" * 70)
print("ШАГ 3: ПОДБОР ОПТИМАЛЬНОГО ПАРАМЕТРА h")
print("=" * 70)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

h_values = [0.1, 0.3, 0.5, 0.7, 1.0, 1.5, 2.0, 3.0, 5.0]
h_scores = []

print("Тестирование h на валидационной выборке (200 объектов):\n")

for h in h_values:
    clf = ParzenWindowClassifier(h=h)
    clf.fit(X_train, y_train)

    sample_idx = np.random.choice(len(X_test), 200, replace=False)
    accuracy = clf.score(X_test[sample_idx], y_test[sample_idx])
    h_scores.append(accuracy)
    print(f"  h = {h:4.1f} → Accuracy = {accuracy:.4f}")

best_h = h_values[np.argmax(h_scores)]
print(f"\nВЫВОД: Оптимальное h = {best_h} (максимальная точность)\n")


print("=" * 70)
```

```
print("ШАГ 4: ТЕСТИРОВАНИЕ НА 10 НЕЗАВИСИМЫХ РАЗБИЕНИЯХ")
print("=" * 70)
print(f"Параметры: h = {best_h}, разбиение 70% обучение / 30% тест\n")

results = []
n_splits = 10

for i in range(n_splits):
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.3, random_state=i, stratify=y
    )

    clf = ParzenWindowClassifier(h=best_h)
    clf.fit(X_train, y_train)

    start_time = time.time()
    y_pred = clf.predict(X_test)
    pred_time = time.time() - start_time

    accuracy = np.mean(y_pred == y_test)
    tp = np.sum((y_pred == 1) & (y_test == 1))
    tn = np.sum((y_pred == 0) & (y_test == 0))
    fp = np.sum((y_pred == 1) & (y_test == 0))
    fn = np.sum((y_pred == 0) & (y_test == 1))

    precision = tp / (tp + fp) if (tp + fp) > 0 else 0
    recall = tp / (tp + fn) if (tp + fn) > 0 else 0
    f1 = 2 * precision * recall / (precision + recall) if (precision +
recall) > 0 else 0

    results.append({
        'Разбиение': i + 1,
        'Размер обуч.': len(X_train),
        'Размер тест.': len(X_test),
        'Accuracy': f"{accuracy:.4f}",
        'Precision': f"{precision:.4f}",
        'Recall': f"{recall:.4f}",
        'F1-Score': f"{f1:.4f}",
        'Время (с)': f"{pred_time:.2f}"
    })

    print(f"Разбиение {i+1:2d}: Accuracy = {accuracy:.4f}, F1 = {f1:.4f}")


results_df = pd.DataFrame(results)

print("\n" + "=" * 70)
print("ТАБЛИЦА РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ")
print("=" * 70)
print(results_df.to_string(index=False))
```

```
accuracies = [float(r['Accuracy']) for r in results]
f1_scores = [float(r['F1-Score']) for r in results]

print("\n" + "=" * 70)
print("ИТОГОВАЯ СТАТИСТИКА")
print("=" * 70)
print(f"Средняя Accuracy:  {np.mean(accuracies):.4f} ±
{np.std(accuracies):.4f}")
print(f"Средний F1-Score:  {np.mean(f1_scores):.4f} ±
{np.std(f1_scores):.4f}")
print(f"Мин. Accuracy:     {np.min(accuracies):.4f}")
print(f"Макс. Accuracy:    {np.max(accuracies):.4f}")
```