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

Дорогин Н.С.

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

Новосибирск 2025 г.

# **Введение (задание)**

Разработка классификатора на основе метода k ближайших соседей.

**Основная часть:**

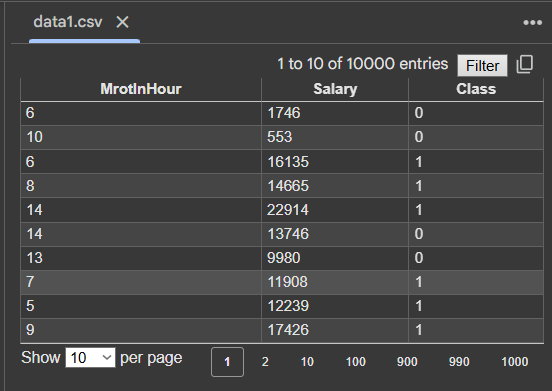
**1.** Загружаем набор данных в соответствии с вариантом:

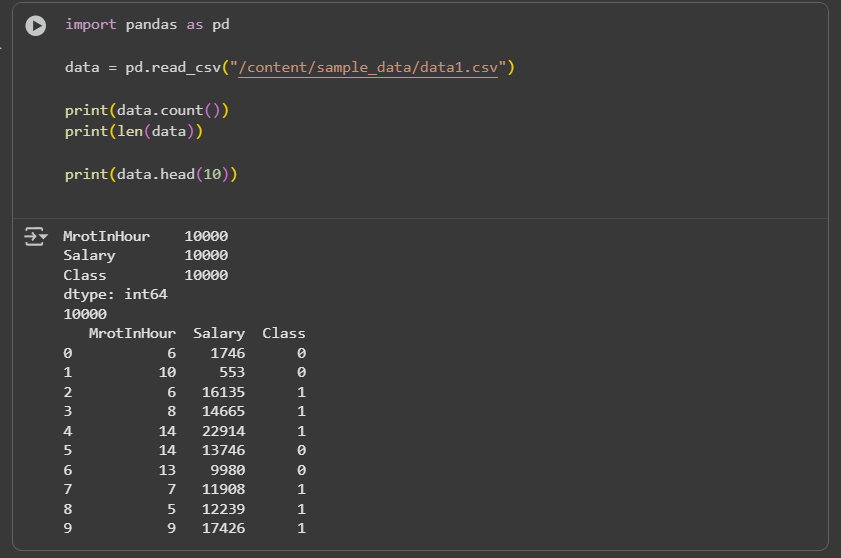
𝑁ф = ((𝑁𝑐 +2) 𝑚𝑜𝑑 5) + 1, где 𝑁ф – номер файла, 𝑁𝑐 − номер по списку

группы.

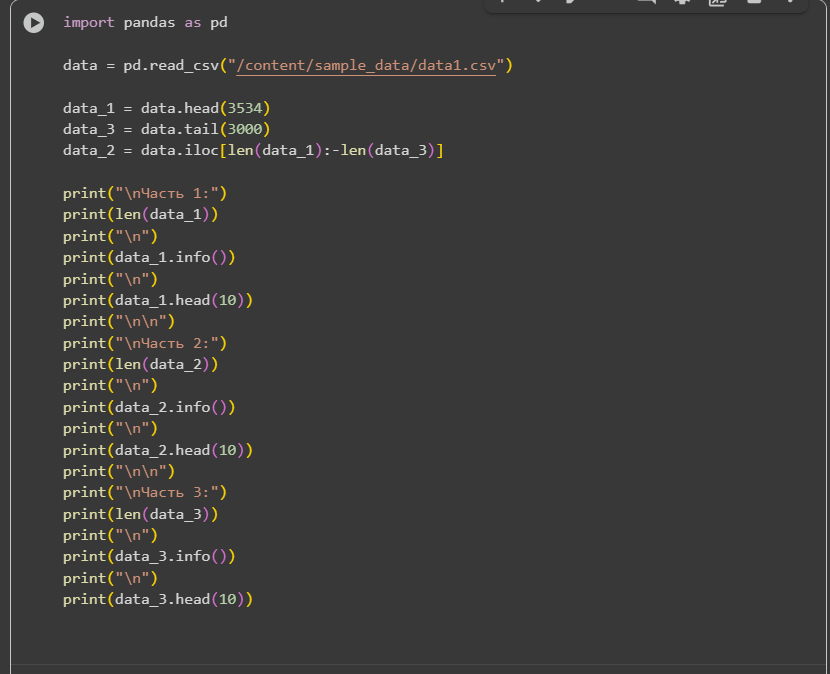
Мой номер в списке - 8.

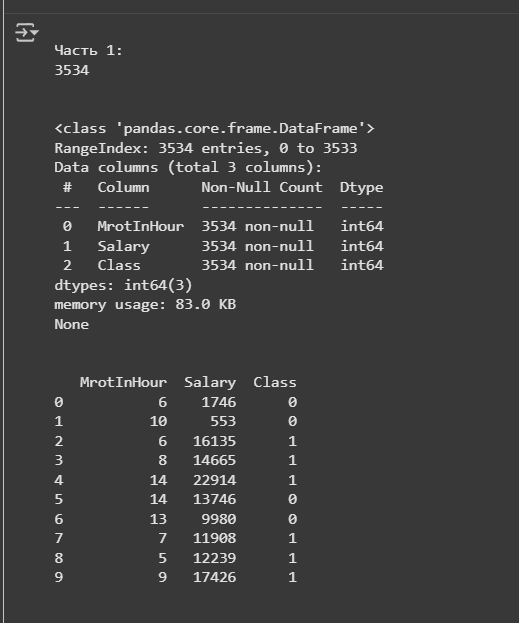
𝑁ф = ((8 + 2) mod 5) + 1 = (10 mod 5) + 1 = 0 + 1 = 1

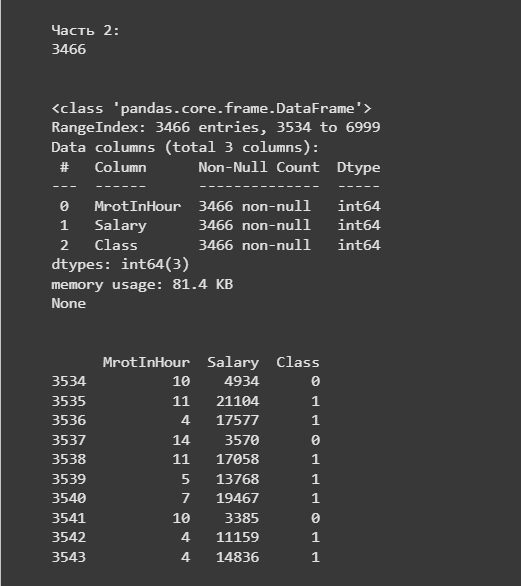


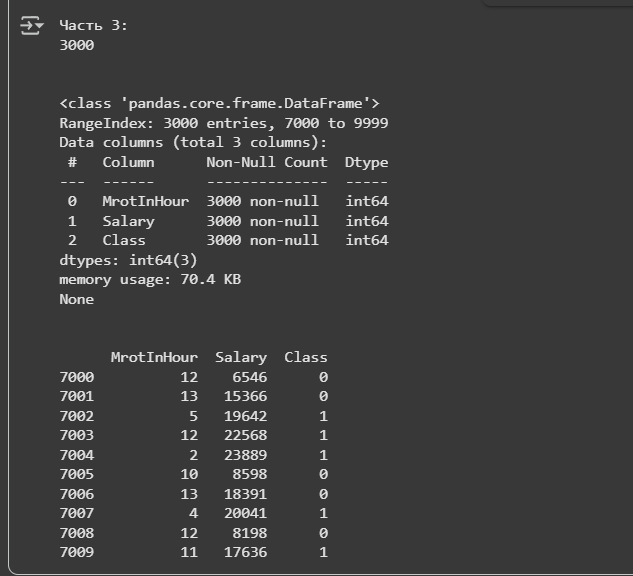


2. Делим выборку на 3 части, первые 2 части используем в качестве обучающей, последнюю - в качестве тестовой. (соотношение обучающей и тестовой 70/30)









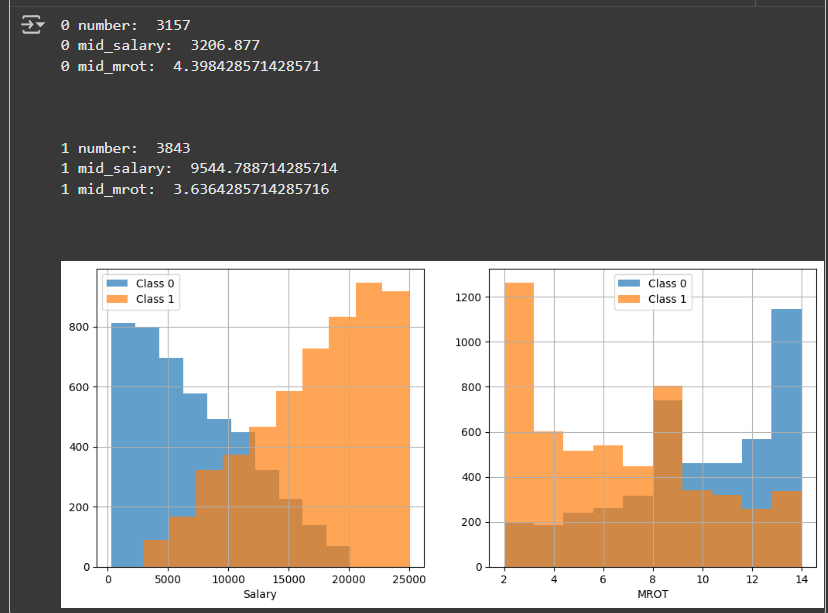
Проанализируем обучающую выборку на возможность минимизировать разницу между количеством представленных в ней объектов разных классов.

Сначала выясним количество уникальных классов:



Всего два уникальных: 0 и 1. Теперь подсчитаем количество и средние показатели каждого.

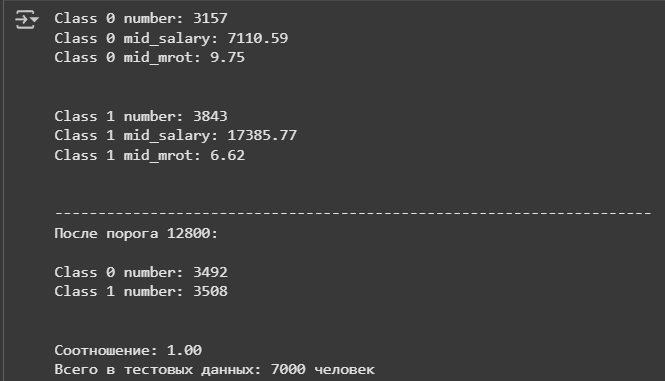
Также по гистограммам посмотрим, насколько показатели пересекаются в целом.



(Вертикальная ось показывает частоту (количество наблюдений) - сколько человек попадает в каждый интервал значений.)

Как можно заметить у двух классов сильное разделение по зарплате и слабое по мрот.

Самое простое, что можно придумать - установить порог зарплаты, при котором мы причислим человека к классу 1 и отсечь таким образом в пользу класса 0 тех, кто ниже этого порога.

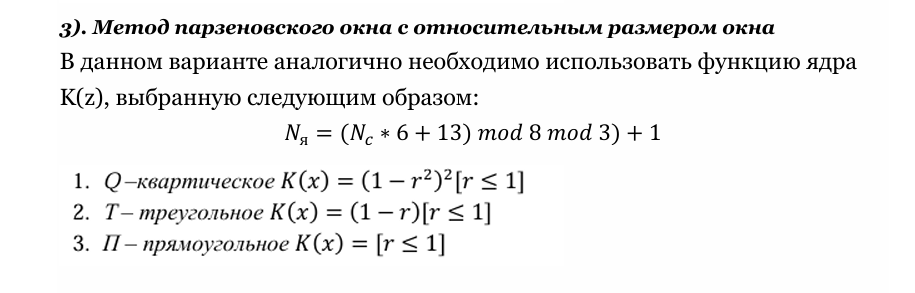


Методом подбора мы определили, что при пороговой заработной плате 12800 соотношение между классами станет практически 1 к 1.

3. 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:

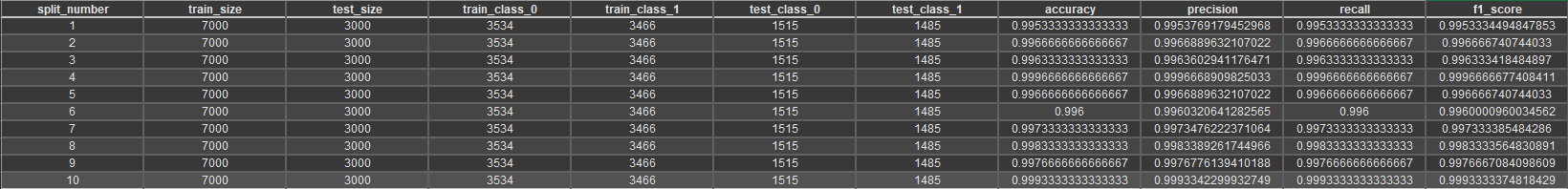
𝑁в =(𝑁𝑐 𝑚𝑜𝑑 3)+1, где 𝑁в – номер варианта, 𝑁𝑐 − номер по списку группы

𝑁в = (8 𝑚𝑜𝑑 3) + 1 = 3



𝑁я = ((8 ∗ 6 + 13) 𝑚𝑜𝑑 8 𝑚𝑜𝑑 3) + 1 = (61 mod 8 mod 3) + 1 = (5 mod 3) + 1 = 2 + 1 = 3

**4. ТАБЛИЦА РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ:**



**5.** [**https://colab.research.google.com/drive/1InYZSFYOxjGShodyJXLZvcIH-hd156SS#scrollTo=PupjpElilssW**](https://colab.research.google.com/drive/1InYZSFYOxjGShodyJXLZvcIH-hd156SS#scrollTo=PupjpElilssW)

# **Код программы:**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

import matplotlib.pyplot as plt

class ParzenWindowClassifier:

    def \_\_init\_\_(self, window\_type='rectangular', relative\_window\_size=0.1):

        self.window\_type = window\_type

        self.relative\_window\_size = relative\_window\_size

        self.X\_train = None

        self.y\_train = None

        self.classes = None

    def \_rectangular\_kernel(self, distances, h):

        """Прямоугольное ядро: K(x) = [r <= 1]"""

        return (distances <= h).astype(float)

    def fit(self, X, y):

        """Обучение классификатора"""

        self.X\_train = X.values if isinstance(X, pd.DataFrame) else X

        self.y\_train = y.values if isinstance(y, pd.Series) else y

        self.classes = np.unique(self.y\_train)

        # Вычисляем ширину окна на основе относительного размера

        data\_range = np.ptp(self.X\_train, axis=0)  # размах данных

        self.h = self.relative\_window\_size \* data\_range.mean()

        return self

    def predict\_proba(self, X):

        """Предсказание вероятностей классов"""

        X\_test = X.values if isinstance(X, pd.DataFrame) else X

        n\_test = X\_test.shape[0]

        n\_classes = len(self.classes)

        probabilities = np.zeros((n\_test, n\_classes))

        for i, x in enumerate(X\_test):

            distances = np.linalg.norm(self.X\_train - x, axis=1)

            weights = self.\_rectangular\_kernel(distances, self.h)

            for j, cls in enumerate(self.classes):

                class\_weights = weights[self.y\_train == cls]

                probabilities[i, j] = np.sum(class\_weights)

            if np.sum(probabilities[i]) > 0:

                probabilities[i] /= np.sum(probabilities[i])

            else:

                probabilities[i] = 1.0 / n\_classes

        return probabilities

    def predict(self, X):

        """Предсказание классов"""

        probabilities = self.predict\_proba(X)

        return self.classes[np.argmax(probabilities, axis=1)]

def evaluate\_classification(y\_true, y\_pred):

    """Полная оценка классификации"""

    accuracy = accuracy\_score(y\_true, y\_pred)

    precision = precision\_score(y\_true, y\_pred, average='weighted', zero\_division=0)

    recall = recall\_score(y\_true, y\_pred, average='weighted', zero\_division=0)

    f1 = f1\_score(y\_true, y\_pred, average='weighted', zero\_division=0)

    return {

        'accuracy': accuracy,

        'precision': precision,

        'recall': recall,

        'f1\_score': f1

    }

def run\_experiment(data, salary\_col, mrot\_col, n\_splits=10, window\_size=0.15):

    """Запуск эксперимента с несколькими разбиениями"""

    # Балансировка классов по порогу зарплаты

    salary\_threshold = 12800

    balanced\_labels = (data[salary\_col] >= salary\_threshold).astype(int)

    features = data[[salary\_col, mrot\_col]]

    results = []

    for split\_num in range(1, n\_splits + 1):

        # Разбиение на обучающую и тестовую выборки (70/30)

        X\_train, X\_test, y\_train, y\_test = train\_test\_split(

            features, balanced\_labels,

            test\_size=0.3,

            random\_state=42 + split\_num,  # разные random\_state для разных разбиений

            stratify=balanced\_labels

        )

        # Обучение классификатора

        classifier = ParzenWindowClassifier(relative\_window\_size=window\_size)

        classifier.fit(X\_train, y\_train)

        # Предсказание

        y\_pred = classifier.predict(X\_test)

        # Оценка качества

        metrics = evaluate\_classification(y\_test, y\_pred)

        # Статистика по разбиению

        split\_info = {

            'split\_number': split\_num,

            'train\_size': len(X\_train),

            'test\_size': len(X\_test),

            'train\_class\_0': sum(y\_train == 0),

            'train\_class\_1': sum(y\_train == 1),

            'test\_class\_0': sum(y\_test == 0),

            'test\_class\_1': sum(y\_test == 1),

        }

        split\_info.update(metrics)

        results.append(split\_info)

        print(f"Разбиение {split\_num}: Точность = {metrics['accuracy']:.4f}")

    return pd.DataFrame(results)

# Основная программа

if \_\_name\_\_ == "\_\_main\_\_":

    # Загрузка данных

    data = pd.read\_csv("/content/sample\_data/data1.csv")

    # Определение имен столбцов

    column\_name = data.columns[2]  # класс

    salary\_col = data.columns[1]   # зарплата

    mrot\_col = data.columns[0]     # МРОТ

    print("=" \* 60)

    print("ЭКСПЕРИМЕНТ С МЕТОДОМ ПАРЗЕНОВСКОГО ОКНА")

    print("=" \* 60)

    print(f"Размер исходных данных: {len(data)} строк")

    print(f"Столбцы: {list(data.columns)}")

    print(f"Балансировка классов по порогу зарплаты: {12800}")

    print()

    # Запуск эксперимента с 10 разбиениями

    results\_df = run\_experiment(data, salary\_col, mrot\_col, n\_splits=10, window\_size=0.15)

    # Вывод подробной таблицы результатов

    print("\n" + "=" \* 80)

    print("ТАБЛИЦА РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ")

    print("=" \* 80)

    # Форматированная таблица результатов

    display\_columns = [

        'split\_number', 'train\_size', 'test\_size',

        'train\_class\_0', 'train\_class\_1', 'test\_class\_0', 'test\_class\_1',

        'accuracy', 'precision', 'recall', 'f1\_score'

    ]

    formatted\_df = results\_df[display\_columns].copy()

    formatted\_df.columns = [

        'Разбиение', 'Обучающая', 'Тестовая',

        'Класс 0 (обуч)', 'Класс 1 (обуч)', 'Класс 0 (тест)', 'Класс 1 (тест)',

        'Точность', 'Точность (prec)', 'Полнота (rec)', 'F1-мера'

    ]

    # Округление метрик

    for col in ['Точность', 'Точность (prec)', 'Полнота (rec)', 'F1-мера']:

        formatted\_df[col] = formatted\_df[col].round(4)

    print(formatted\_df.to\_string(index=False))

    # Статистика по всем разбиениям

    print("\n" + "=" \* 60)

    print("СТАТИСТИКА ПО ВСЕМ РАЗБИЕНИЯМ")

    print("=" \* 60)

    metrics = ['accuracy', 'precision', 'recall', 'f1\_score']

    stats = {}

    for metric in metrics:

        values = results\_df[metric]

        stats[metric] = {

            'mean': values.mean(),

            'std': values.std(),

            'min': values.min(),

            'max': values.max()

        }

    stats\_df = pd.DataFrame(stats).T.round(4)

    stats\_df.columns = ['Среднее', 'Стд. откл.', 'Минимум', 'Максимум']

    print(stats\_df)

    # Визуализация результатов

    plt.figure(figsize=(12, 8))

    # График точности по разбиениям

    plt.subplot(2, 2, 1)

    plt.plot(results\_df['split\_number'], results\_df['accuracy'], 'bo-', linewidth=2, markersize=8)

    plt.xlabel('Номер разбиения')

    plt.ylabel('Точность')

    plt.title('Точность классификации по разбиениям')

    plt.grid(True, alpha=0.3)

    # График распределения классов в обучающей выборке

    plt.subplot(2, 2, 2)

    width = 0.35

    x = np.arange(len(results\_df))

    plt.bar(x - width/2, results\_df['train\_class\_0'], width, label='Класс 0', alpha=0.7)

    plt.bar(x + width/2, results\_df['train\_class\_1'], width, label='Класс 1', alpha=0.7)

    plt.xlabel('Номер разбиения')

    plt.ylabel('Количество объектов')

    plt.title('Распределение классов в обучающей выборке')

    plt.legend()

    plt.grid(True, alpha=0.3)

    # Boxplot метрик

    plt.subplot(2, 2, 3)

    metrics\_data = [results\_df['accuracy'], results\_df['precision'],

                   results\_df['recall'], results\_df['f1\_score']]

    plt.boxplot(metrics\_data, labels=['Точность', 'Precision', 'Recall', 'F1-score'])

    plt.title('Распределение метрик качества')

    plt.grid(True, alpha=0.3)

    # График всех метрик

    plt.subplot(2, 2, 4)

    plt.plot(results\_df['split\_number'], results\_df['accuracy'], 'bo-', label='Точность')

    plt.plot(results\_df['split\_number'], results\_df['precision'], 'ro-', label='Precision')

    plt.plot(results\_df['split\_number'], results\_df['recall'], 'go-', label='Recall')

    plt.plot(results\_df['split\_number'], results\_df['f1\_score'], 'mo-', label='F1-score')

    plt.xlabel('Номер разбиения')

    plt.ylabel('Значение метрики')

    plt.title('Все метрики по разбиениям')

    plt.legend()

    plt.grid(True, alpha=0.3)

    plt.tight\_layout()

    plt.show()

    # Сохранение результатов в файл

    results\_df.to\_csv('parzen\_window\_results.csv', index=False)

    print(f"\nРезультаты сохранены в файл: parzen\_window\_results.csv")

    # Итоговый вывод

    print("\n" + "=" \* 60)

    print("ИТОГОВЫЕ ВЫВОДЫ")

    print("=" \* 60)

    print(f"Средняя точность: {stats\_df.loc['accuracy', 'Среднее']:.4f} ± {stats\_df.loc['accuracy', 'Стд. откл.']:.4f}")

    print(f"Лучшая точность: {stats\_df.loc['accuracy', 'Максимум']:.4f}")

    print(f"Худшая точность: {stats\_df.loc['accuracy', 'Минимум']:.4f}")

    print(f"Стабильность классификатора: {1 - stats\_df.loc['accuracy', 'Стд. откл.']:.4f}")