

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.

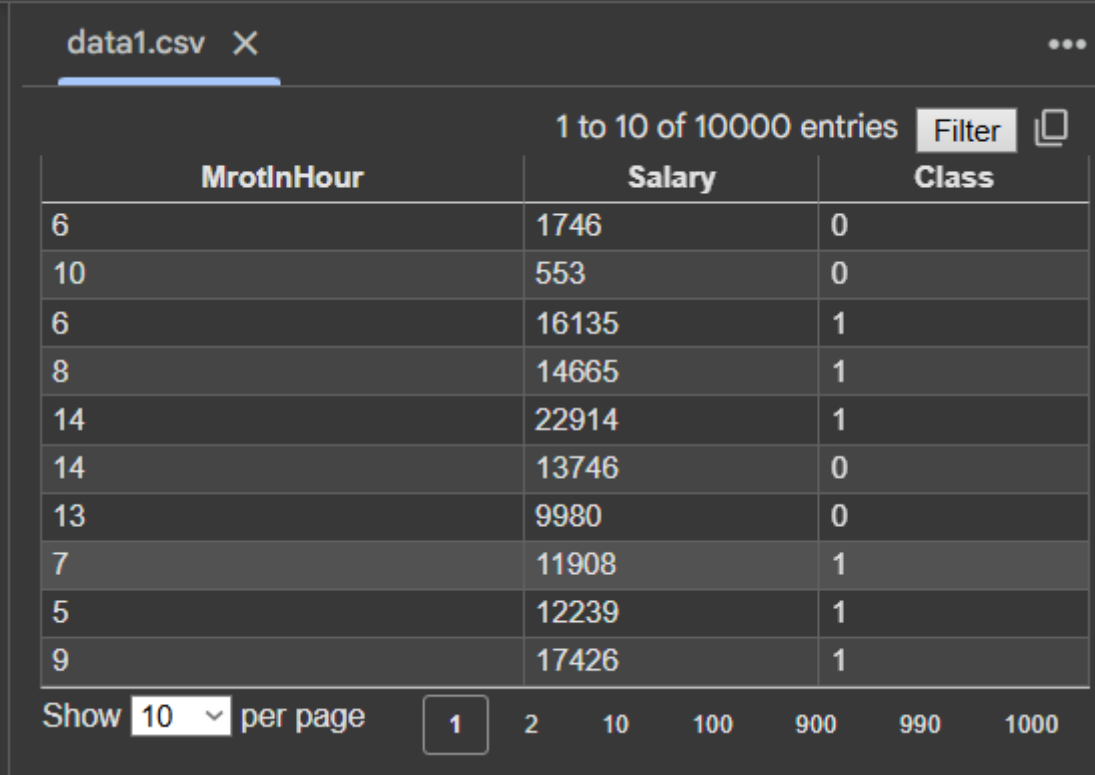
Основная часть:

1. Загружаем набор данных в соответствии с вариантом:

$N_{\text{ф}} = ((N_{\text{с}} + 2) \bmod 5) + 1$, где $N_{\text{ф}}$ – номер файла, $N_{\text{с}}$ – номер по списку группы.

Мой номер в списке - 8.

$$N_{\text{ф}} = ((8 + 2) \bmod 5) + 1 = (10 \bmod 5) + 1 = 0 + 1 = 1$$



1 to 10 of 10000 entries Filter		
MrotInHour	Salary	Class
6	1746	0
10	553	0
6	16135	1
8	14665	1
14	22914	1
14	13746	0
13	9980	0
7	11908	1
5	12239	1
9	17426	1

Show 10 per page

1 2 10 100 900 990 1000

```
import pandas as pd

data = pd.read_csv("/content/sample_data/data1.csv")

print(data.count())
print(len(data))

print(data.head(10))
```

```
⇒ MrotInHour    10000
   Salary       10000
   Class        10000
dtype: int64
10000
```

	MrotInHour	Salary	Class
0	6	1746	0
1	10	553	0
2	6	16135	1
3	8	14665	1
4	14	22914	1
5	14	13746	0
6	13	9980	0
7	7	11908	1
8	5	12239	1
9	9	17426	1

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

```
import pandas as pd

data = pd.read_csv("/content/sample_data/data1.csv")

data_1 = data.head(3534)
data_3 = data.tail(3000)
data_2 = data.iloc[len(data_1):-len(data_3)]

print("\nЧасть 1:")
print(len(data_1))
print("\n")
print(data_1.info())
print("\n")
print(data_1.head(10))
print("\n\n")
print("\nЧасть 2:")
print(len(data_2))
print("\n")
print(data_2.info())
print("\n")
print(data_2.head(10))
print("\n\n")
print("\nЧасть 3:")
print(len(data_3))
print("\n")
print(data_3.info())
print("\n")
print(data_3.head(10))
```



Часть 1:
3534

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 3534 entries, 0 to 3533  
Data columns (total 3 columns):  
#   Column      Non-Null Count  Dtype  
---  -  
0   MrotInHour  3534 non-null   int64  
1   Salary      3534 non-null   int64  
2   Class       3534 non-null   int64  
dtypes: int64(3)  
memory usage: 83.0 KB  
None
```

	MrotInHour	Salary	Class
0	6	1746	0
1	10	553	0
2	6	16135	1
3	8	14665	1
4	14	22914	1
5	14	13746	0
6	13	9980	0
7	7	11908	1
8	5	12239	1
9	9	17426	1

Часть 2:
3466

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 3466 entries, 3534 to 6999  
Data columns (total 3 columns):  
#   Column      Non-Null Count  Dtype  
---  -  
0   MrotInHour  3466 non-null   int64  
1   Salary      3466 non-null   int64  
2   Class       3466 non-null   int64  
dtypes: int64(3)  
memory usage: 81.4 KB  
None
```

	MrotInHour	Salary	Class
3534	10	4934	0
3535	11	21104	1
3536	4	17577	1
3537	14	3570	0
3538	11	17058	1
3539	5	13768	1
3540	7	19467	1
3541	10	3385	0
3542	4	11159	1
3543	4	14836	1



Часть 3:

3000

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 7000 to 9999
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  -
0   MrotInHour  3000 non-null   int64
1   Salary      3000 non-null   int64
2   Class       3000 non-null   int64
dtypes: int64(3)
memory usage: 70.4 KB
None
```

	MrotInHour	Salary	Class
7000	12	6546	0
7001	13	15366	0
7002	5	19642	1
7003	12	22568	1
7004	2	23889	1
7005	10	8598	0
7006	13	18391	0
7007	4	20041	1
7008	12	8198	0
7009	11	17636	1

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

Сначала выясним количество уникальных классов:

```
[102]
✓ 0
сек.

import pandas as pd

data = pd.read_csv("/content/sample_data/data1.csv")
column_name = data.columns[2]

data_1 = data.head(3534)
data_3 = data.tail(3000)
data_2 = data.iloc[len(data_1):-len(data_3)]

classes = []

for i, row in data_1.iterrows():
    if row[column_name] not in classes:
        classes.append(row[column_name])

for i, row in data_2.iterrows():
    if row[column_name] not in classes:
        classes.append(row[column_name])

print(classes)
```

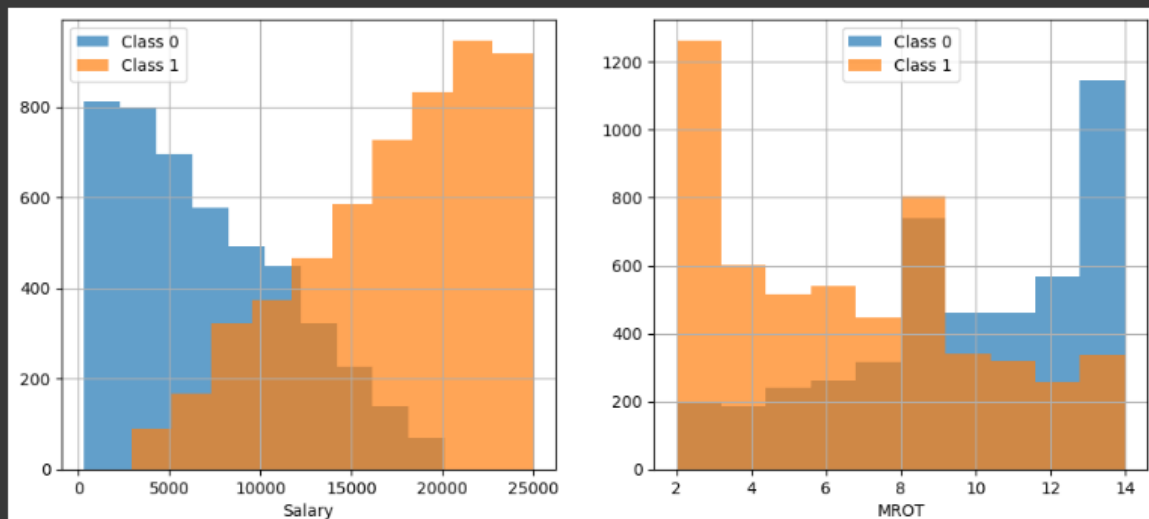
```
→ [np.int64(0), np.int64(1)]
```

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

Также по гистограммам посмотрим, насколько показатели пересекаются в целом.

```
0 number: 3157
0 mid_salary: 3206.877
0 mid_mrot: 4.398428571428571

1 number: 3843
1 mid_salary: 9544.788714285714
1 mid_mrot: 3.6364285714285716
```



(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.


```
→ Class 0 number: 3157  
Class 0 mid_salary: 7110.59  
Class 0 mid_mrot: 9.75
```

```
Class 1 number: 3843  
Class 1 mid_salary: 17385.77  
Class 1 mid_mrot: 6.62
```

После порога 12800:

```
Class 0 number: 3492  
Class 1 number: 3508
```

```
Соотношение: 1.00  
Всего в тестовых данных: 7000 человек
```

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:

$N_{\text{в}} = (N_{\text{с}} \bmod 3) + 1$, где $N_{\text{в}}$ – номер варианта, $N_{\text{с}}$ – номер по списку группы

$$N_{\text{в}} = (8 \bmod 3) + 1 = 3$$

3). Метод парзеновского окна с относительным размером окна

В данном варианте аналогично необходимо использовать функцию ядра $K(z)$, выбранную следующим образом:

$$N_{\text{я}} = (N_{\text{с}} * 6 + 13) \bmod 8 \bmod 3 + 1$$

1. Q –квартическое $K(x) = (1 - r^2)^2 [r \leq 1]$
2. T –треугольное $K(x) = (1 - r) [r \leq 1]$
3. P –прямоугольное $K(x) = [r \leq 1]$

$$Nя = ((8 * 6 + 13) \bmod 8 \bmod 3) + 1 = (61 \bmod 8 \bmod 3) + 1 = (5 \bmod 3) + 1 = 2 + 1 = 3$$

4. ТАБЛИЦА РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ:

split_number	train_size	test_size	train_class_0	train_class_1	test_class_0	test_class_1	accuracy	precision
1	7000	3000	3534	3466	1515	1485	0.9953333333333333	0.9953769179452968
2	7000	3000	3534	3466	1515	1485	0.9966666666666667	0.9966889632107022
3	7000	3000	3534	3466	1515	1485	0.9963333333333333	0.9963602941176471
4	7000	3000	3534	3466	1515	1485	0.9996666666666667	0.999668909825033
5	7000	3000	3534	3466	1515	1485	0.9966666666666667	0.9966889632107022
6	7000	3000	3534	3466	1515	1485	0.996	0.9960320641282565
7	7000	3000	3534	3466	1515	1485	0.9973333333333333	0.9973476222371064
8	7000	3000	3534	3466	1515	1485	0.9983333333333333	0.9983389261744966
9	7000	3000	3534	3466	1515	1485	0.9976666666666667	0.9976776139410188
10	7000	3000	3534	3466	1515	1485	0.9993333333333333	0.9993342299932749

5. <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',
'Mаксимум']
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',
'Sреднее']:.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}")
```