Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

Факультет	**Программной Инженерии и Компьютерной Техники**
Направление подготовки (специальность)	**Системное и прикладное программное обеспечение**
Дисциплина	**Системы искусственного интеллекта**

ЛАБОРАТОРНАЯ РАБОТА 4
ОТЧЕТ

Выполнил студент:	**Силинцев Владислав Витальевич (355273)**
Группа:	**P3314**
Преподаватель:	**Болдырева Елена Александровна (157150)**

г. Санкт-Петербург

2025

Содержание

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ....3
ОТЧЕТ О ХОДЕ ВЫПОЛНЕНИЯ....4
Выбор датасета....4
Визуализация статистики....4
Обработка данных....6
Метод k-ближайших соседей....6
Построение моделей....7
Оценка производительности....8
Матрицы ошибок....9
Разработанное приложение....10
ЗАКЛЮЧЕНИЕ....12

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

- Выбор датасетов:
    - Студенты с четным порядковым номером в группе должны использовать набор данных о вине.
    - Студенты с нечетным порядковым номером в группе должны использовать про диабет.
- Проведите предварительную обработку данных, включая обработку отсутствующих значений, кодирование категориальных признаков и масштабирование.
- Получите и визуализируйте (графически) статистику по датасету (включая количество, среднее значение, стандартное отклонение, минимум, максимум и различные квантили), постройте 3d-визуализацию признаков.
- Реализуйте метод k-ближайших соседей без использования сторонних библиотек, кроме NumPy и Pandas.
- Постройте две модели k-NN с различными наборами признаков:
    - Модель 1: Признаки случайно отбираются.
    - Модель 2: Фиксированный набор признаков, который выбирается заранее.
- Для каждой модели проведите оценку на тестовом наборе данных при разных значениях k. Выберите несколько различных значений k, например, k=3, k=5, k=10, и т. д. Постройте матрицу ошибок.

ОТЧЕТ О ХОДЕ ВЫПОЛНЕНИЯ

Выбор датасета

В соответствии с порядковым номером в группе (8 — чётное число) для выполнения работы был выбран набор данных WineDataset, содержащий информацию о вине.

Визуализация статистики

Для анализа распределения и взаимосвязей признаков были построены:

- Гистограммы распределения каждого признака
- Box-plot по каждому признаку для анализа выбросов
- 3D-визуализация признаков

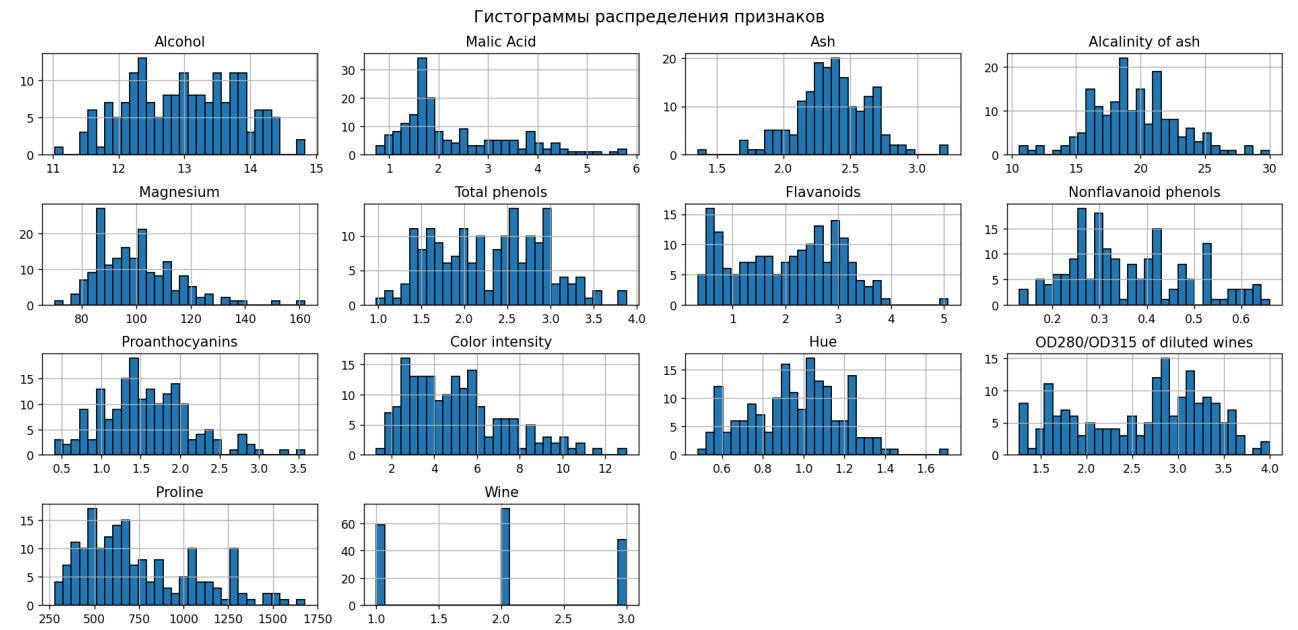

Рисунок 1 – Гистограммы распределения признаков.

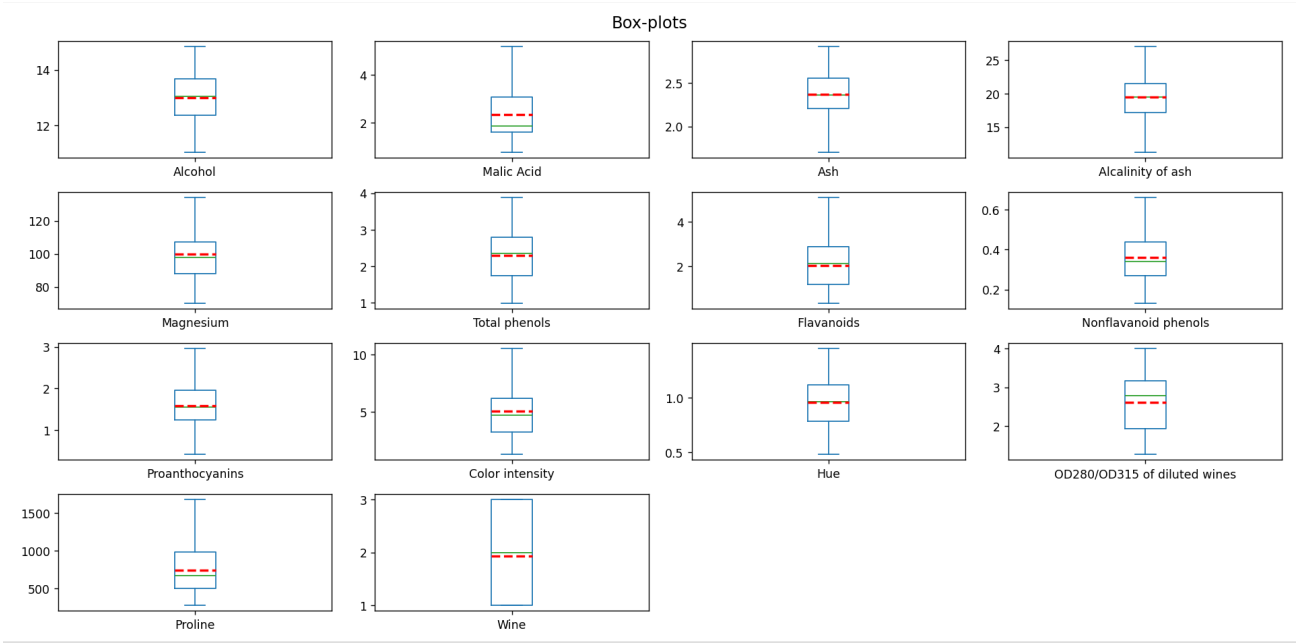

Рисунок 2 – Box-plot по каждому признаку.

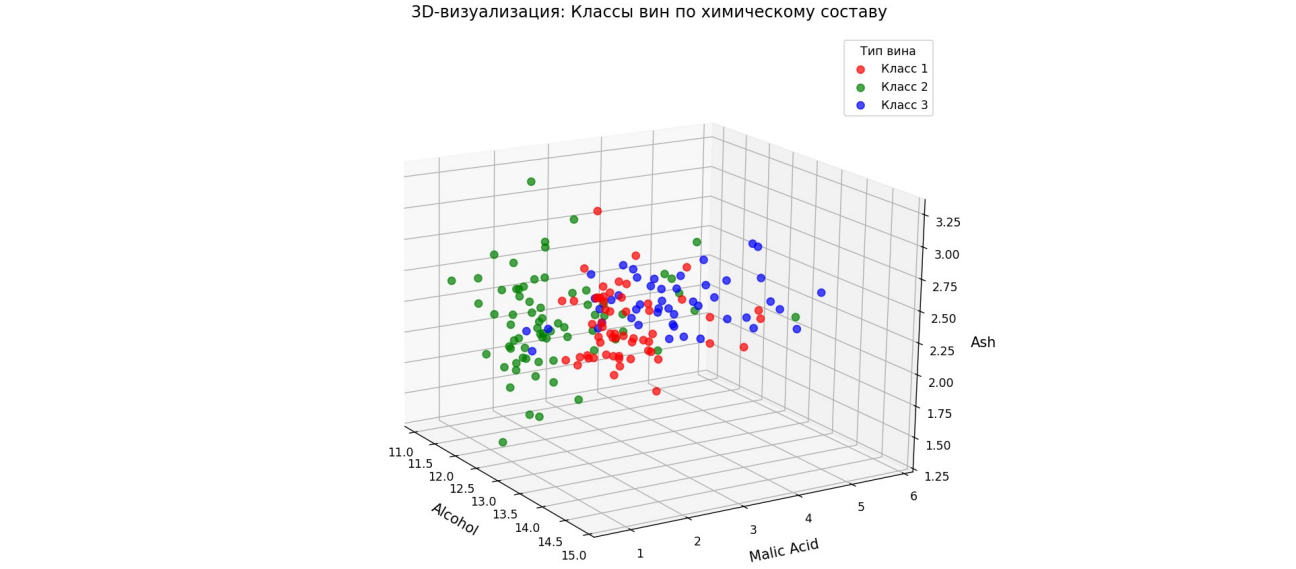

Рисунок 3 – 3D-визуализация признаков.

Обработка данных

```
# Загрузка CSV
pd.set_option('display.max_columns', None)
df = pd.read_csv(get_resource_path("WineDataset.csv"))

df = df.dropna()  # Убрать пустые строки
```

Метод k-ближайших соседей

```
import numpy as np
import pandas as pd
from collections import Counter


def calculate_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))


class KNNClassifier:
    """
    Реализация метода k-ближайших соседей для классификации
    """

    def __init__(self, k=3):
        self.k = k
        self.X_train = None
        self.y_train = None

    def fit(self, x, y):
        """Сохранение обучающих данных"""
        # Преобразуем в numpy массивы для единообразия
        self.X_train = np.array(x)
        self.y_train = np.array(y)
        return self

    def predict(self, x_test):
        """Предсказание классов для новых данных"""
        x_test = np.array(x_test)
        y_predicted = np.zeros(x_test.shape[0], dtype=self.y_train.dtype)

        # Для каждой точки из тестового набора
        for i, test_point in enumerate(x_test):
            # 1. Вычисляем расстояния до всех обучающих точек
            distances = []
            for j, train_point in enumerate(self.X_train):
                dist = calculate_distance(test_point, train_point)
```

```
                distances.append((dist, self.y_train[j]))

            # 2. Сортируем по расстоянию (от ближайшего к дальнему)
            distances.sort(key=lambda x: x[0])

            # 3. Выбираем k ближайших соседей
            k_nearest = distances[:self.k]

            # 4. Извлекаем метки классов соседей
            k_nearest_labels = [label for _, label in k_nearest]

            # 5. Выбираем наиболее частый класс (режим)
            most_common = Counter(k_nearest_labels).most_common(1)[0][0]
            y_predicted[i] = most_common
        return y_predicted
```

Построение моделей

```
# Модель 1: случайные
random.seed(42)
rand_feat = random.sample([c for c in df.columns if c != 'Wine'], 3)
X1 = df[rand_feat].values
X1_norm = (X1 - X1.mean(0)) / (X1.std(0) + 1e-10)
X1_train, X1_test = train_test_split_custom(X1_norm, y, test_size=0.2)[:2]
print("МОДЕЛЬ 1 (случайные признаки):")

# Матрицы для модели 1
cms1 = []
accs1 = []
for k in k_values:
    knn = KNNClassifier(k=k)
    knn.fit(X1_train, y_train)
    y_pred = knn.predict(X1_test)
    acc = np.mean(y_pred == y_test)
    cm = cm3(y_test, y_pred)

    cms1.append(cm)
    accs1.append(acc)

    print(f"k={k}: {acc:.2%} | Матрица: {cm}")

print("=" * 100)

# Модель 2: фиксированные
fix_feat = ['Alcohol', 'Malic Acid', 'Ash']
X2 = df[fix_feat].values
```

```
X2_norm = (X2 - X2.mean(0)) / (X2.std(0) + 1e-10)
X2_train, X2_test = train_test_split_custom(X2_norm, y, test_size=0.2)[:2]
print("МОДЕЛЬ 2 (фиксированные признаки):")

# Матрицы для модели 2
cms2 = []
accs2 = []
for k in k_values:
    knn = KNNClassifier(k=k)
    knn.fit(X2_train, y_train)
    y_pred = knn.predict(X2_test)
    acc = np.mean(y_pred == y_test)
    cm = cm3(y_test, y_pred)

    cms2.append(cm)
    accs2.append(acc)

    print(f"k={k}: {acc: .2%} | Матрица: {cm}")
```

Оценка производительности

```
# Создаем фигуры
fig1, axes1 = plt.subplots(1, len(k_values), figsize=(15, 3))
fig2, axes2 = plt.subplots(1, len(k_values), figsize=(15, 3))

# Модель 1: все матрицы в одной строке
for i, (k, cm, acc) in enumerate(zip(k_values, cms1, accs1)):
    plot_cm(axes1[i] if len(k_values) > 1 else axes1, cm, k, acc)
    if i == 0:
        axes1[i].set_ylabel('Истинный класс', fontsize=10)
    axes1[i].set_xlabel('Предсказанный', fontsize=10)

fig1.suptitle('Модель 1: Матрицы ошибок для разных k', fontsize=14)

# Модель 2: все матрицы в одной строке
for i, (k, cm, acc) in enumerate(zip(k_values, cms2, accs2)):
    plot_cm(axes2[i] if len(k_values) > 1 else axes2, cm, k, acc)
    if i == 0:
        axes2[i].set_ylabel('Истинный класс', fontsize=10)
    axes2[i].set_xlabel('Предсказанный', fontsize=10)

fig2.suptitle('Модель 2: Матрицы ошибок для разных k', fontsize=14)

plt.tight_layout()
plt.show()
```

Матрицы ошибок

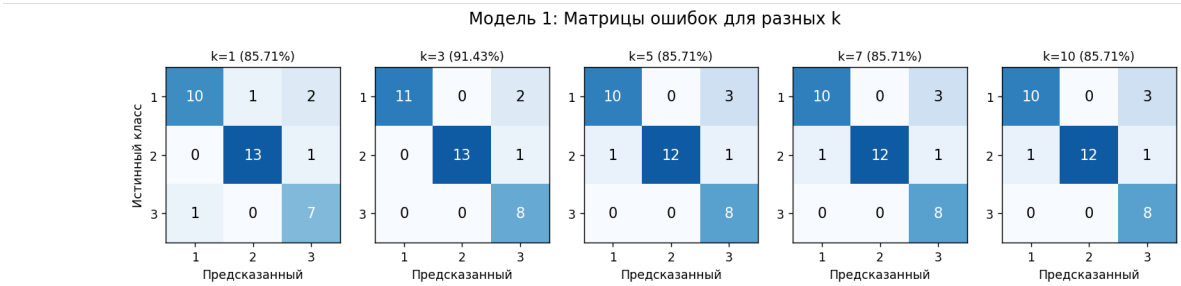

Рисунок 4.1 – Матрицы ошибок для модели 1.

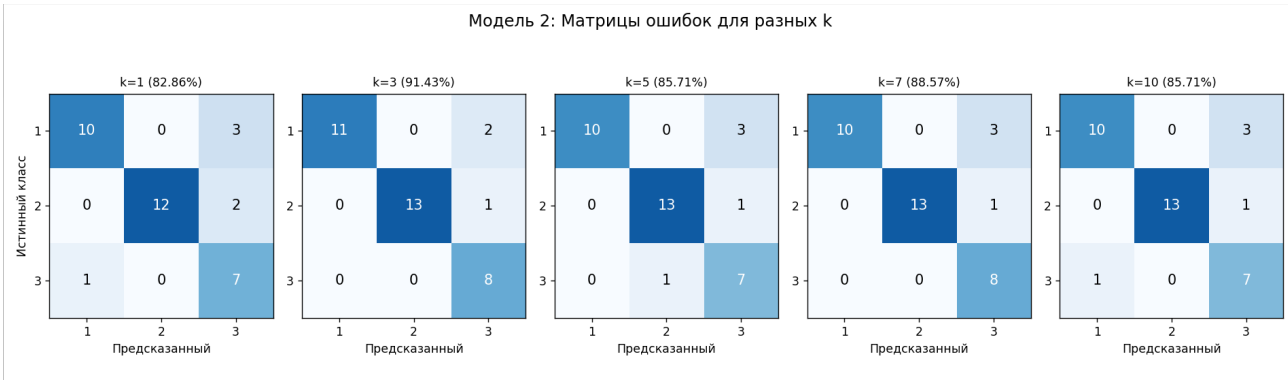

Рисунок 4.2 – Матрицы ошибок для модели 2.

Разработанное приложение

Исходный код приложения: https://github.com/vvlaads/AI-systems-4.

Пример работы приложения:

```
================================================================================================
Основная статистика:
          Alcohol  Malic Acid         Ash  Alcalinity of ash   Magnesium  \
count  178.000000  178.000000  178.000000         178.000000  178.000000
mean    13.000618    2.336348    2.366517          19.494944   99.741573
std      0.811827    1.117146    0.274344           3.339564   14.282484
min     11.030000    0.740000    1.360000          10.600000   70.000000
25%     12.362500    1.602500    2.210000          17.200000   88.000000
50%     13.050000    1.865000    2.360000          19.500000   98.000000
75%     13.677500    3.082500    2.557500          21.500000  107.000000
max     14.830000    5.800000    3.230000          30.000000  162.000000

       Total phenols  Flavanoids  Nonflavanoid phenols  Proanthocyanins  \
count     178.000000  178.000000            178.000000       178.000000
mean        2.295112    2.029270              0.361854         1.590899
std         0.625851    0.998859              0.124453         0.572359
min         0.980000    0.340000              0.130000         0.410000
25%         1.742500    1.205000              0.270000         1.250000
50%         2.355000    2.135000              0.340000         1.555000
75%         2.800000    2.875000              0.437500         1.950000
max         3.880000    5.080000              0.660000         3.580000
```

Рисунок 4.1 – Пример работы приложения.

```
       Color intensity         Hue  OD280/OD315 of diluted wines      Proline  \
count       178.000000  178.000000                    178.000000   178.000000
mean          5.058090    0.957449                      2.611685   746.893258
std           2.318286    0.228572                      0.709990   314.907474
min           1.280000    0.480000                      1.270000   278.000000
25%           3.220000    0.782500                      1.937500   500.500000
50%           4.690000    0.965000                      2.780000   673.500000
75%           6.200000    1.120000                      3.170000   985.000000
max          13.000000    1.710000                      4.000000  1680.000000

             Wine
count  178.000000
mean     1.938202
std      0.775035
min      1.000000
25%      1.000000
50%      2.000000
75%      3.000000
max      3.000000
================================================================================================
Размеры выборок:
  Обучающая: 143 образцов
  Тестовая:  35 образцов
================================================================================================
```

Рисунок 4.2 – Пример работы приложения.

```
МОДЕЛЬ 1 (случайные признаки):
k=1: 85.71% | Матрица: [[10, 1, 2], [0, 13, 1], [1, 0, 7]]
k=3: 91.43% | Матрица: [[11, 0, 2], [0, 13, 1], [0, 0, 8]]
k=5: 85.71% | Матрица: [[10, 0, 3], [1, 12, 1], [0, 0, 8]]
k=7: 85.71% | Матрица: [[10, 0, 3], [1, 12, 1], [0, 0, 8]]
k=10: 85.71% | Матрица: [[10, 0, 3], [1, 12, 1], [0, 0, 8]]
================================================================================================
МОДЕЛЬ 2 (фиксированные признаки):
k=1:  82.86% | Матрица: [[10, 0, 3], [0, 12, 2], [1, 0, 7]]
k=3:  91.43% | Матрица: [[11, 0, 2], [0, 13, 1], [0, 0, 8]]
k=5:  85.71% | Матрица: [[10, 0, 3], [0, 13, 1], [0, 1, 7]]
k=7:  88.57% | Матрица: [[10, 0, 3], [0, 13, 1], [0, 0, 8]]
k=10:  85.71% | Матрица: [[10, 0, 3], [0, 13, 1], [1, 0, 7]]
```

Рисунок 4.3 – Пример работы приложения.

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы был изучен и применен метод k-ближайших соседей для классификации вин на основе химических показателей. Были реализованы две модели: с использованием случайно отобранных признаков и с фиксированным набором характеристик.

Проведен анализ влияния параметра k на точность классификации, построены матрицы ошибок для значений k от 1 до 10. Результаты показали, что метод KNN эффективно решает поставленную задачу, демонстрируя высокую точность при правильно подобранном количестве соседей (оптимально 3–7).