федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт информационных и вычислительных технологий
Кафедра управления и интеллектуальных технологий

Отчёт по лабораторной работе №3
По дисциплине «Методы и алгоритмы обработки данных и изображений»

Выполнили студенты: Михайловский М. Ю., Озеров С. Д.
Группа: А-02м-25
Бригада: 2
Проверил: Бородкин А. А.

Москва 2025

Содержание

1 ВЫПОЛНЕНИЕ РАБОТЫ

1.1 Формирование выборки векторизованных голосовых команд

В качестве исходных данных были взяты записи набора голосовых команд: **«вперед, назад, налево, направо, стоп, разворот, вверх, вниз, увеличить, уменьшить, сканировать, осмотреться»**. У каждой команды было 10 отдельных записей. Итого получается выборка из 120 голосовых команд.

В качестве набора информативных признаков голосовой команды были выбраны усреднённые мелчастотные кестральные коэффициенты (mfcc) по набору кадров длиной 10 мс из исходной записи голосовой команды. Примеры визуализаций векторизованных команд приведены на рис. 1.1-1.2.

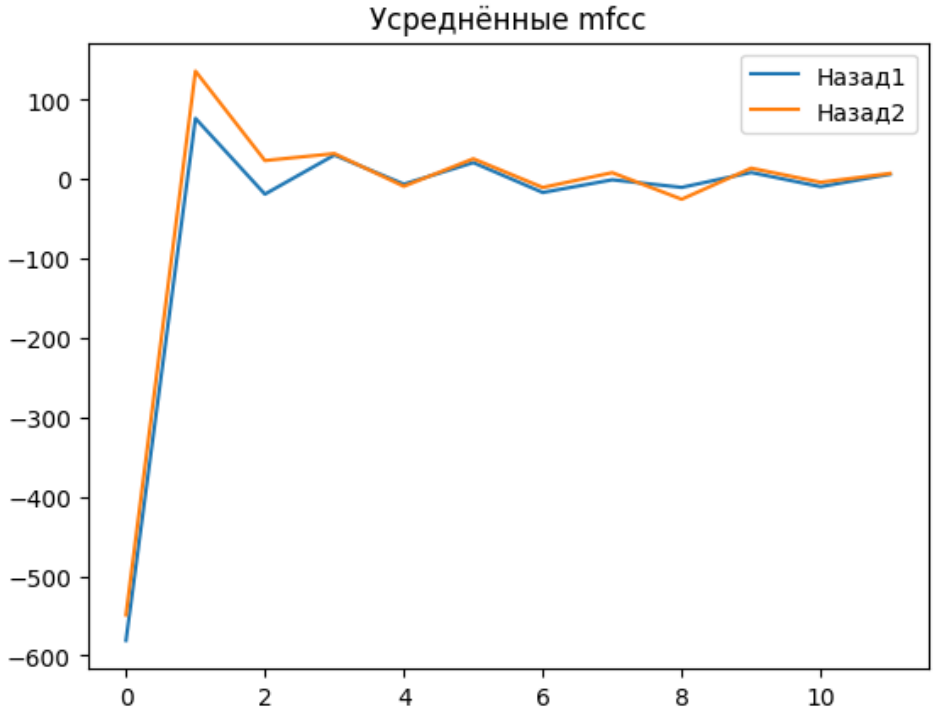

Рис. 1.1. Усреднённые mfcc для двух одинаковых команд «назад»

Визуально различия графики для разных команд имеют похожий характер, однако по ним может быть проведена классификация.

Для дальнейшей классификации были выделены тренировочная и тестовая выборки в соотношении 4 к 1.

1.2 Классификация с помощью DTW

Метод динамической трансформации шкалы времени (DTW) позволяет получить путь деформации между двумя голосовыми командами. Он представляет собой минимальный путь. Длина пути деформации, по сути, представляет собой некую меру близости двух голосовых команд.

Для классификации рассчитывается длина пути деформации между классифицируемой голосовой командой и всеми голосовыми командами из тренировочной выборки. Ей присваивается метка голосовой команды ближайшей по данной метрике. Классификатор был реализован в виде класса, представленного на листинге 3.

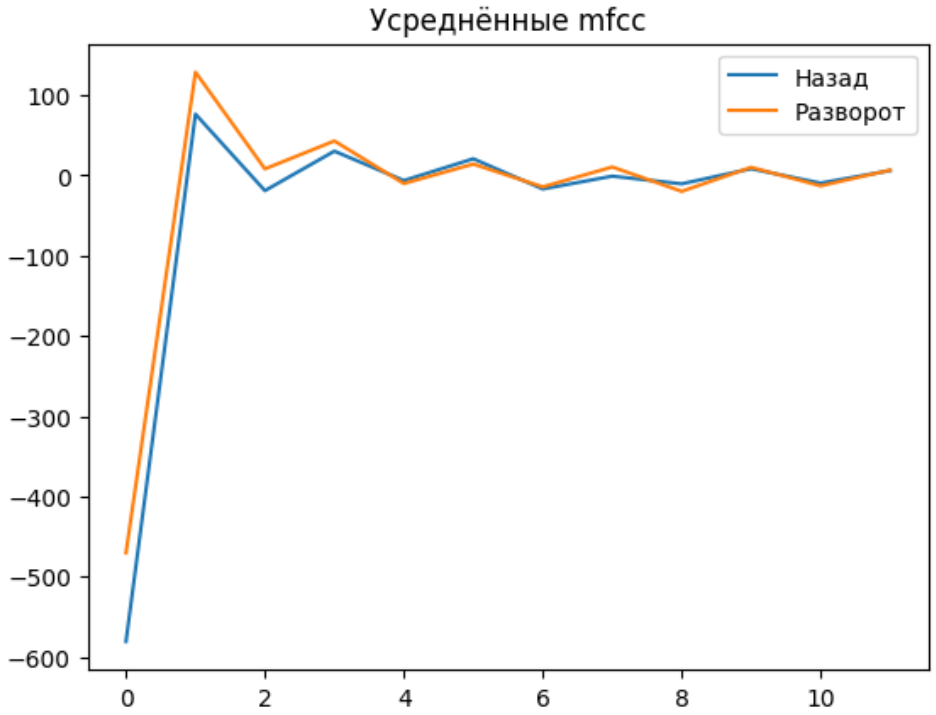

Рис. 1.2. Усреднённые mfcc для двух разных команд «назад» и «разворот»

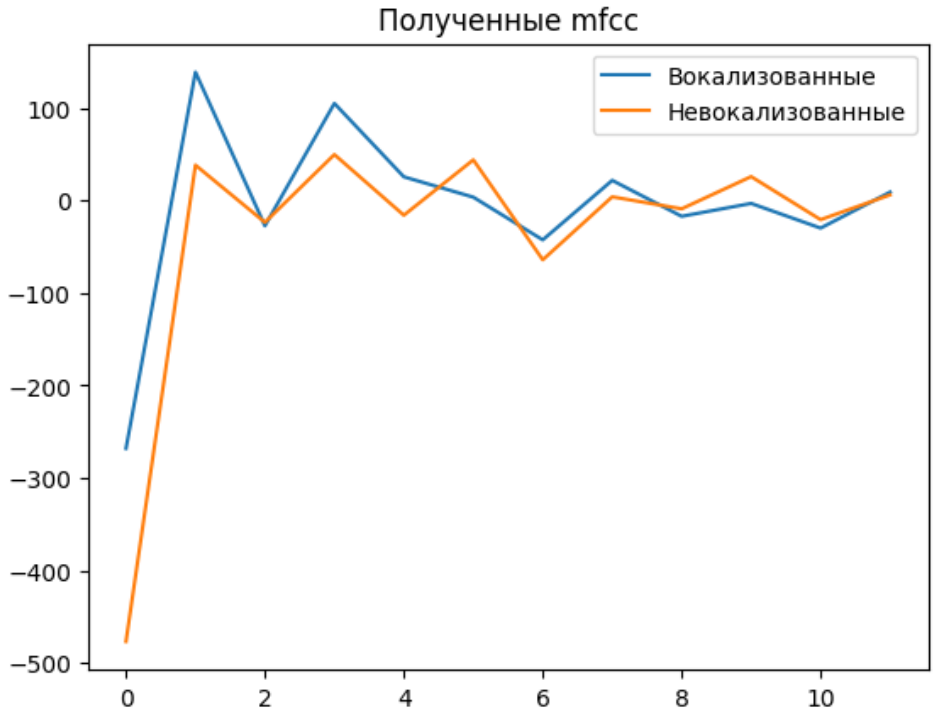

Рис. 1.3. Усреднённые mfcc для вокализованного участка и не вокализованного

По своей сути, реализованный классификатор на основе DTW это метод 1-го ближайшего соседа с использованием длины пути деформации в качестве метрики расстояния.

1.3 Классификация с помощью многослойного персептрона

Классификатор был реализован в виде класса NN, представленного в листинге 4. В своей структуре это персептрон с двумя скрытыми слоями с 32 и 16 нейронами соответственно

и функциями активации RELU и LeakyRELU соответственно. На выходе используется 12 нейронов с функцией активации softmax, т.к. производится классификация 12 голосовых команд.

Обучение производится с помощью алгоритма оптимизации adam с использованием категориальной кросс-энтропии в качестве минимизируемой метрики. Для обучения из тренировочной выборки дополнительно было выделено 20% голосовых команд в валидационную выборку. Обучение производилось до момента времени, пока функция потери на валидационной выборке не станет расти в течение 3 эпох подряд.

1.4 Полученные результаты

Для сравнения возможностей данных классификаторов в разных условиях был реализован класс Pipeline, описанный в листинге 5. Он обеспечивал общий интерфейс взаимодействия с обоими классификаторами.

Затем была написана программа, представленная в листинге 6, которая обучила ряд классификаторов в различных условиях и сформировала сводную таблицу (листинг 1). По этой таблице были построены графики 1.4-1.7.

Листинг 1. Сводная таблица обученных классификаторов голосовых команд

№	Степень перекрытия	Кол-во инф. признаков	DTW точность	NN точность	DTW время, мс	NN время, мс
1	0.25	6	0.791667	0.541667	208.1284	164.0788
2	0.25	12	0.791667	0.916667	554.6691	162.7194
3	0.25	18	0.875000	0.833333	916.7622	188.6156
4	0.50	6	0.791667	0.583333	220.4520	188.1650
5	0.50	12	0.791667	0.833333	599.4209	183.5267
6	0.50	18	0.833333	0.916667	979.2463	166.5307
7	0.75	6	0.708333	0.583333	191.3799	164.2450
8	0.75	12	0.791667	0.875000	520.1233	424.2866
9	0.75	18	0.875000	0.916667	831.2984	165.6572

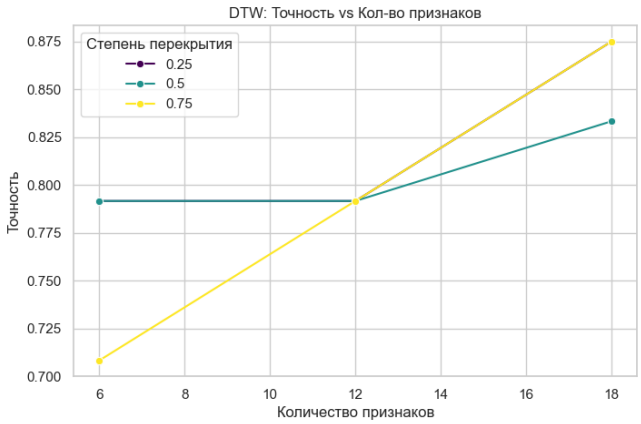

Рис. 1.4. Графики зависимости точности DTW от условий

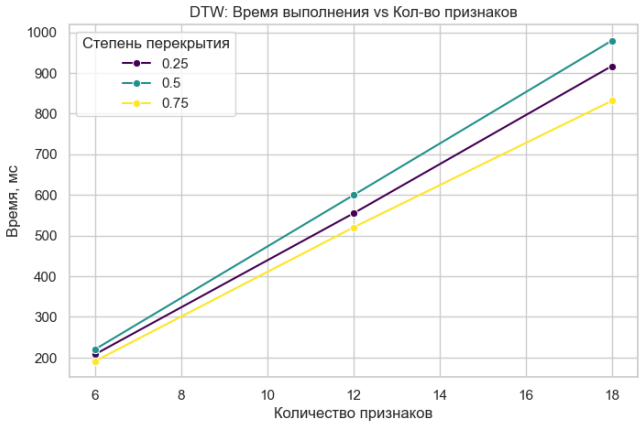

Рис. 1.5. Графики зависимости быстродействия DTW от условий

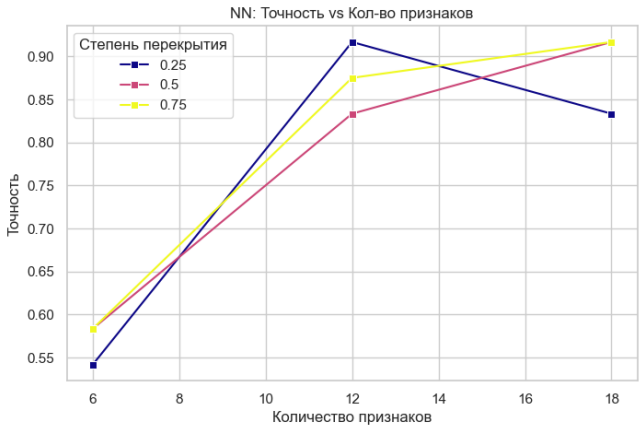

Рис. 1.6. Графики зависимости точности NN от условий

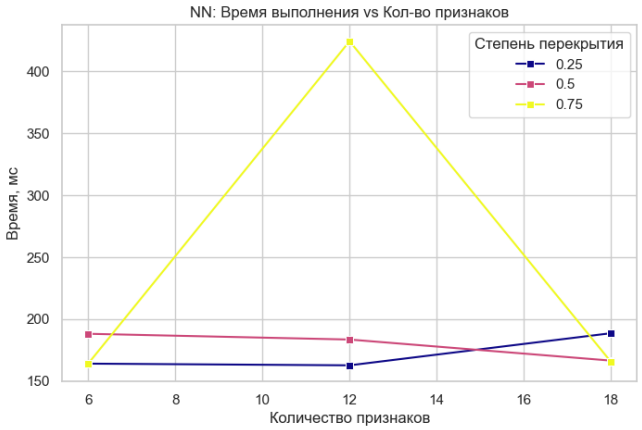

Рис. 1.7. Графики зависимости быстродействия NN от условий

Увеличение степени перекрытия или количества признаков сказывается на быстродействии, но может оказать хорошее воздействие на точности. Для нейросетевого классификатора эти графики не совсем однозначные, поскольку итоговая точность классификатора сильно зависит от исходных весов нейросети, и на практике варьировалась в диапазоне 10%.

Однако стоит отметить, что наиболее критичным было малое количество признаков для итоговой точности.

2 ВЫВОДЫ

В данной лабораторной работе были реализованы два различных классификатора голосовых команд. Один основан на методе динамической трансформации шкалы времени (DTW), и был реализован на основе метода вида 1-го ближайшего соседа. Второй классификатор был реализован в виде многослойного персептрона.

Были обучено 18 классификаторов для сравнения их показателей в различных условиях.

Приложение А. Листинги

Листинг 2. Векторизация голосовых команд и визуализация mfcc

```
import os
import soundfile as sf
import librosa
import numpy as np
import matplotlib.pyplot as plt

soundfiles_path = './notebook/'
command_folders = {command : soundfiles_path + command for command in
↪  os.listdir(soundfiles_path)}
individual_commands_folders = {}

for command, command_folder in command_folders.items():
        files = os.listdir(command_folder)
        file_paths = []
        for file in files:
        #file_paths = [f'{command_folder}/{file}' for file in files ]
        numbers = set('1234567890')
        symbols_in_file = set(file)
        numbers_in_filename = set.intersection(numbers, symbols_in_file)

        if not numbers_in_filename:
                continue

        file_path = f'{command_folder}/{file}'
        file_paths.append(file_path)

        individual_commands_folders[command] = file_paths

print(command_folders)
print(individual_commands_folders)

# 1 график
back1_avg = x_avg[33]
back2_avg = x_avg[35]
plt.plot(back1_avg, label='Назад1')
plt.plot(back2_avg, label='Назад2')
plt.legend()
plt.title('Усреднённые mfcc')
plt.show()

# 2 график
around_avg = x_avg[75]
plt.plot(back1_avg, label='Назад')
plt.plot(around_avg, label='Разворот')
plt.legend()
plt.title('Усреднённые mfcc')
plt.show()

vowel_path = './vowel_scan1.wav'
consonant_path = './consonant_scan1.wav'
```

```
50
51 # 3 график
52 vowel_mfcc = np.mean(calc_mfcc(vowel_path), axis=1)
53 consonant_mfcc = np.mean(calc_mfcc(consonant_path), axis=1)
54
55 plt.plot(vowel_mfcc, label='Вокализованные')
56 plt.plot(consonant_mfcc, label='Невокализованные')
57 plt.legend()
58 plt.title('Полученные mfcc')
59 plt.show()
```

Листинг 3. Классификатор DTW

```
1  from sklearn.neighbors import KNeighborsClassifier
2  from sklearn.model_selection import train_test_split
3  from fastdtw import fastdtw
4  from tqdm import tqdm
5
6  class DTWClassifier:
7          def __init__(self):
8                  pass
9
10         @staticmethod
11         def dtw_metric(x, y):
12                 return fastdtw(x, y)[0]
13
14         def fit(self, train_values, targets):
15                 self.train_values = train_values
16                 self.targets = targets
17
18         def predict(self, value):
19                 min_metric = np.inf
20                 min_metric_i = -1
21                 for i, train_value in enumerate(self.train_values):
22                         cur_metric = DTWClassifier.dtw_metric(train_value, value)
23                         if cur_metric < min_metric:
24                                 min_metric = cur_metric
25                                 min_metric_i = i
26
27                 return self.targets[min_metric_i]
28
29         def score(self, values, targets):
30                 total = len(values)
31                 correct = 0
32                 for value, target in tqdm(list(zip(values, targets))):
33                         predicted_target = self.predict(value)
34                         if predicted_target == target:
35                                 correct += 1
36
37                 accuracy = correct / total
38                 return accuracy
```

Листинг 4. Нейросетевой классификатор

```
from tensorflow.keras.regularizers import l2
import tensorflow as tf
from sklearn.preprocessing import LabelEncoder

class NN:
        def initialize_nn_model(self, x_train):
                norm = tf.keras.layers.Normalization(axis=-1)
                norm.adapt(np.array(x_train))

                self.model = tf.keras.models.Sequential([
                        tf.keras.Input(shape=(np.array(x_train).shape[1],)),
                        norm,
                        tf.keras.layers.Dense(32, activation='relu',
                        ↪  kernel_regularizer=l2(1e-5)),
                        tf.keras.layers.Dense(16,
                        ↪  activation=tf.keras.layers.LeakyReLU(alpha=0.01),
                        ↪  kernel_regularizer=l2(1e-5)),
                        tf.keras.layers.Dense(12, activation='softmax')
                ])

                self.model.compile(optimizer='adam',
                        loss='sparse_categorical_crossentropy',
                        metrics=['accuracy']
                )


        def fit(self, x_train, y_train):
                x_train = np.array(x_train)
                y_train = np.array(y_train)
                self.initialize_nn_model(x_train)

                early_stop = tf.keras.callbacks.EarlyStopping(
                        monitor='val_loss',
                        patience=3,
                        restore_best_weights=False
                )

                self.le = LabelEncoder()
                y_train_enc = self.le.fit_transform(y_train)
                self.model.fit(np.array(x_train),
                        np.array(y_train_enc),
                        epochs=1000,
                        validation_split=0.2,
                        callbacks=[early_stop]
                )

        def evaluate(self, x, y):
                y = np.array(y)
                y_enc = self.le.transform(y)
                return self.model.evaluate(np.array(x), np.array(y_enc))

        def predict(self, x):
                x = np.array(x)
                return self.model.predict(x)
```

```
52
53          def score(self, x, y):
54                  return self.evaluate(x, y)[1]
```

Листинг 5. Pipeline для общего интерфейса работы с обоими классификаторами

```
1  from time import perf_counter
2
3  class Pipeline:
4          def __init__(self, classifier=DTWClassifier, intersection=0.5, n_mfcc=12):
5                  self.classifier = classifier()
6                  self.intersection = intersection
7                  self.n_mfcc = n_mfcc
8
9          def fit(self, x, targets):
10                 self.classifier.fit(x, targets)
11
12         def predict(self, x):
13                 return self.classifier.predict(x)
14
15         def score(self, x, targets):
16                 return self.classifier.score(x, targets)
17
18         def generate_fit_score(self):
19                 _, targets, x_avg = calc_mfccs_from_files(individual_commands_folders,
                   ↪  intersection=self.intersection, n_mfcc=self.n_mfcc)
20                 x_train, x_test, y_train, y_test = get_train_test(x_avg, targets)
21
22                 self.fit(x_train, y_train)
23
24                 score_start = perf_counter()
25                 accuracy = self.score(x_test, y_test)
26                 score_end = perf_counter()
27
28                 time_spent_ms = (score_end - score_start) * 1000
29
30                 return accuracy, time_spent_ms
31
```

Листинг 6. Тесты качества классификаторов

```
1  import pandas as pd
2  pd.set_option('display.expand_frame_repr', False)  # отключить перенос строк
3
4  def get_train_test(x, targets, test_size=0.2, random_state=42):
5          max_len = 0
6          for x_elem in x:
7                  cur_len = len(x_elem)
8                  if cur_len > max_len:
9                          max_len = cur_len
10
11         x_unified = []
```

```
        for x_elem in x:
                need_zeros = max_len - len(x_elem)
                if need_zeros:
                        x_unified.append(np.pad(x_elem, (0, need_zeros), mode='constant'))
                else:
                        x_unified.append(x_elem)
        return train_test_split(x_unified, targets, test_size=test_size,
        ↪  random_state=random_state)

x_train, x_test, y_train, y_test = get_train_test(x_avg, targets)

# Списки значений параметров
intersections = [0.25, 0.5, 0.75]    # Степень перекрытия
n_mfcc_list = [6, 12, 18]            # Кол-во признаков MFCC

# Пустой список для хранения результатов
results = []

# Перебор комбинаций
idx = 1
for intersection in intersections:
        for n_mfcc in n_mfcc_list:
                # DTW
                pipeline_dtw = Pipeline(DTWClassifier, intersection=intersection, n_mfcc=n_mfcc)
                acc_dtw, time_dtw = pipeline_dtw.generate_fit_score()

                # NN
                pipeline_nn = Pipeline(NN, intersection=intersection, n_mfcc=n_mfcc)
                acc_nn, time_nn = pipeline_nn.generate_fit_score()

                # Добавляем строку в список
                results.append({
                        "№": idx,
                        "Степень перекрытия": intersection,
                        "Кол-во инф. признаков": n_mfcc,
                        "DTW точность": acc_dtw,
                        "NN точность": acc_nn,
                        "DTW время, мс": time_dtw,
                        "NN время, мс": time_nn
                })
                idx += 1

# Создаем DataFrame
df = pd.DataFrame(results)

# Смотрим результат
print(df)
```

Листинг 7. Графики метрик классификаторов

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
4
5  # Предположим, df - твой датафрейм
6  sns.set(style="whitegrid")
7
8  # --- График точности DTW ---
9  plt.figure(figsize=(8,5))
10         sns.lineplot(data=df,
11         x="Кол-во инф. признаков",
12         y="DTW точность",
13         hue="Степень перекрытия",
14         marker="o",
15         palette="viridis")  # разные цвета для разных степеней перекрытия
16 plt.title("DTW: Точность vs Кол-во признаков")
17 plt.xlabel("Количество признаков")
18 plt.ylabel("Точность")
19 plt.legend(title="Степень перекрытия")
20 plt.show()
21
22 # --- График точности NN ---
23 plt.figure(figsize=(8,5))
24 sns.lineplot(data=df,
25         x="Кол-во инф. признаков",
26         y="NN точность",
27         hue="Степень перекрытия",
28         marker="s",
29         palette="plasma")
30 plt.title("NN: Точность vs Кол-во признаков")
31 plt.xlabel("Количество признаков")
32 plt.ylabel("Точность")
33 plt.legend(title="Степень перекрытия")
34 plt.show()
35
36 # --- График времени DTW ---
37 plt.figure(figsize=(8,5))
38 sns.lineplot(data=df,
39         x="Кол-во инф. признаков",
40         y="DTW время, мс",
41         hue="Степень перекрытия",
42         marker="o",
43         palette="viridis")
44 plt.title("DTW: Время выполнения vs Кол-во признаков")
45 plt.xlabel("Количество признаков")
46 plt.ylabel("Время, мс")
47 plt.legend(title="Степень перекрытия")
48 plt.show()
49
50 # --- График времени NN ---
51 plt.figure(figsize=(8,5))
52 sns.lineplot(data=df,
53         x="Кол-во инф. признаков",
54         y="NN время, мс",
55         hue="Степень перекрытия",
56         marker="s",
57         palette="plasma")
58 plt.title("NN: Время выполнения vs Кол-во признаков")
```

```
59 plt.xlabel("Количество признаков")
60 plt.ylabel("Время, мс")
61 plt.legend(title="Степень перекрытия")
62 plt.show()
```