НИТУ «МИСИС»

Институт ИТКН
Кафедра инженерной кибернетики
Направление подготовки: 01.03.04 прикладная математика
Квалификация (степень): бакалавр
Группа: БПМ-21-3
Курс обучения: **3**

ОТЧЕТ

ПО КУРСОВОЙ НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

на тему:

«Приложение для обучающих курсов с применением адаптивного обучения»

6-й семестр, 2023-2024 уч. год

Учащийся: _____ / Хлобустова С.М. /
подпись Фамилия И.О.

Оценка: Отлично

Дата защиты: 06.06.2024

Руководитель КНИР _____ / _____ /
подпись должность, уч., степень. Фамилия И.О.

Москва 2024

Оглавление

1. Список сокращений

CAT (Computer-Adaptive Testing, компьютерно-адаптивное тестирование) - это метод проведения тестов, который динамически адаптируется к уровню успеваемости испытуемого, изменяя сложность представленных заданий в соответствии с предыдущими ответами испытуемого.

RL (Reinforcement learning) - обучение с подкреплением, где происходит изучение того, как агенты должны принимать последовательные решения в среде для максимизации некоторой кумулятивной награды. В отличие от обучения с учителем, где модель обучается на фиксированном наборе данных, обучение с подкреплением предполагает взаимодействие агента с динамической средой и получение обратной связи в виде наград.

MDP (Markov Decision Process, марковский процесс принятия решений) - это случайный процесс, который описывает эволюцию системы с течением времени, когда система переходит из одного состояния в другое вероятностным образом, основываясь на действиях, предпринимаемых лицом, принимающим решения.

PPO (Proximal Policy Optimization, алгоритм ближайшей оптимизации политики) - алгоритм обучения с подкреплением, который сосредотачивается на обучении стратегии (policy) с учетом границ для обновлений.

IRT (Item Response Theory, современная теория тестирования) - набор методов, позволяющий оценить вероятность правильного ответа испытуемых на задания различной трудности.

A2C (Advantage Actor-Critic) — это алгоритм обучения с подкреплением, который относится к классу методов актер-критик. Этот метод сочетает в себе элементы полисного градиента и временного различия для улучшения стабильности и эффективности обучения агентов.

TD (Temporal Difference) - это метод обучения в машинном обучении, который используется для оценки функции ценности (value function) в задачах с подкреплением (Reinforcement Learning, RL). Метод TD сочетает в себе идеи

динамического программирования и Монте-Карло методов, предоставляя эффективный способ обновления оценок ценности состояний или действий на основе временных различий между последовательными оценками.

SARSA (State-Action-Reward-State-Action) - метод, который оценивает ценности действий (Q-функции) и обновляет их на основе наблюдаемой последовательности состояния, действия, награды, следующего состояния и следующего действия.

2. Введение

Проект представляет собой приложение для обучающих курсов, использующее адаптивное обучение. Адаптивное обучение — это персонализированный подход к обучению, который подстраивается под индивидуальные потребности каждого учащегося. В рамках проекта была разработана система автоматической оценки, которая оценивает текущий уровень знаний учащегося и предлагает тестовые задания, направленные на улучшение его знаний.

В современном образовательном процессе часто наблюдается отсутствие индивидуального подхода к обучению. Традиционные методы обучения не всегда учитывают уровень подготовки и уникальные потребности каждого учащегося, что приводит к неэффективному усвоению материала. Адаптивное обучение решает эту проблему, позволяя создавать индивидуальные траектории обучения. Система автоматической оценки знаний, которая является центральным элементом КНИР, предоставляет возможность эффективно и точно определять пробелы в знаниях и предлагать задачи, способствующие их устранению. Это повышает качество образования и способствует более глубокому усвоению материала.

Цель проекта - разработка системы автоматической оценки, которая оценивает уровень знаний учащегося и подбирает соответствующие тестовые задания для улучшения его знаний.

В рамках КНИР будут решаться следующие задачи:

- Оценка текущих знаний учащегося на основе его ответов на ранее предложенные вопросы;
- Оценка вероятности правильного ответа учащегося на конкретное задание, используя текущий уровень знаний и характеристики задания;
- Создание и поддержание набора тестовых заданий, из которых система будет выбирать наиболее подходящие вопросы для каждого учащегося;

- Разработка алгоритма, который будет выбирать следующий наиболее информативный элемент на основе данных модели ответов.

Для достижения основной цели и решения задач КНИР были использованы следующие научно-технические подходы:

- IRT используется для изучения характеристик всех тестовых заданий и базовых качеств знаний учащихся. Модель позволяет точно оценить, какие задания являются наиболее подходящими для определения уровня знаний учащегося.
- MDP применяется для нахождения оптимальной политики, представляющей собой функцию, сопоставляющую состояния с действиями. Эта политика максимизирует ожидаемое совокупное вознаграждение (в данном контексте - улучшение знаний учащегося) с течением времени. Методы динамического программирования, такие как итерация значений и итерация политики, используются для итеративного обновления оценок значений каждого состояния.

3. Основная часть

3.1. Математическая постановка

Основные компоненты компьютерного адаптивного тестирования (CAT):

1. Средство оценки уровня знаний:

Функция E оценивает текущие знания учащегося на основе его ответов на ранее предложенные вопросы. Пусть K_i обозначает уровень знаний учащегося i. Тогда $K_i = E(Q_i, A_i)$, где Q_i - множество вопросов, предложенных учащемуся i, и A_i- множество ответов учащегося i.

2. Модель ответа:

Функция P оценивает вероятность правильного ответа учащегося на конкретное задание, используя текущий уровень знаний и характеристики задания. Пусть G_i обозначает вероятность правильного ответа учащегося i на задание j. Тогда $G_i = P(K_i, X_j)$, где X_j - характеристики задания j.

3. Пул доступных заданий:

Множество T представляет собой набор тестовых заданий, из которых адаптивная система выбирает вопросы. Пусть $T = \{X_1, X_2, \dots, X_n\}$.

4. Алгоритм выбора элемента:

Алгоритм S выбирает следующий наиболее информативный элемент на основе выходных данных модели ответов. Пусть Q_{next} обозначает следующее задание, которое будет предложено учащемуся. Тогда $Q_{next} = S(\{P(K_i, X_j)\})$.

Применение IRT и MDP:

- Модель IRT (Item Response Theory):

Используется для изучения характеристик всех тестовых заданий и базовых качеств знаний учащихся. Пусть β_i обозначает параметр сложности задания j, а α_i - дискриминационный параметр. Модель IRT оценивает, насколько задание j подходит для оценки знаний учащегося i.

- MDP (Markov Decision Process):

Процесс Маркова используется для нахождения оптимальной политики π_i, которая представляет собой функцию, сопоставляющую состояния s с действиями a, максимизируя ожидаемое совокупное вознаграждение с течением времени. Пусть $V^{\pi}(s)$ обозначает значение состояния s при следовании политике π_i. Цель - максимизация $V^{\pi}(s)$.

Формальные определения:

1. Функция оценки уровня знаний:

$$K_i = E(Q_i, A_i)$$

2. Модель ответа:

$$G_i = P(K_i, X_j)$$

3. Алгоритм выбора элемента:

$$Q_{next} = S(\{P(K_i, X_j)\})$$

4. Модель IRT:

$$P(X_j | K_i) = \frac{1}{1 + \exp\left(-\alpha_j\left(K_i - \beta_j\right)\right)}$$

5. Процесс Маркова:

$$V^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^{t} r(s_t, a_t) \mid s_0 = s, \pi]$$

где γ - коэффициент дисконтирования, $r(s_t, a_t)$ - вознаграждение за действие a_t в состоянии s_t.

3.2. Основные этапы реализации

1. Подготовка данных и окружения:

1.1 Чтение данных о студентах и тестовых заданиях с помощью функции read_dataset.

1.2 Создание нового окружения SelectionEnv, которое будет использоваться для обучения и тестирования модели.

1.3 Проверка корректности созданного окружения с помощью специальной функции проверки.

2. Парсинг аргументов командной строки:

2.1 Определение необходимых аргументов, таких как количество тестовых студентов, количество тестов, максимальное количество эпизодов, флаг для вывода дополнительной информации, настройки гиперпараметров и директория для сохранения результатов.

2.2 Чтение и обработка аргументов командной строки для дальнейшего использования в программе.

3. Гиперпараметрическая оптимизация:

3.1 Определение диапазонов значений для гиперпараметров, таких как имя политики, коэффициент дисконтирования, коэффициент энтропии, скорость обучения и количество проходов.

3.2 Перебор всех возможных комбинаций заданных гиперпараметров и выполнение обучения модели для каждой комбинации.

3.3 Пропуск комбинации гиперпараметров, если результаты для этой комбинации уже существуют, если флаг force_repeat не установлен.

4. Обучение и тестирование модели:

4.1 Обучение модели PPO с использованием заданного окружения и гиперпараметров. Обучение проводится для определенного количества эпизодов.

4.2 После обучения модель тестируется на новых данных (тестовых студентах). В процессе тестирования вычисляется дисконтированный доход для каждого студента.

5. Сохранение результатов:

5.1 Сохранение дисконтированного дохода и других результатов тестирования в JSON-файл в указанной директории.

5.2 Если установлен флаг intermediate, результаты обучения сохраняются после каждого прохода для последующего анализа.

6. Дополнительные функции:

6.1 Тестирование различных методов окружения SelectionEnv для проверки их корректной работы.

6.2 Создание нового экземпляра окружения с заданными параметрами, такими как идентификаторы студентов, количество тестовых студентов, выходные данные, количество тестов и максимальное количество эпизодов.

7. Основной поток выполнения:

7.1 Основная функция main запускается при выполнении скрипта, обеспечивая выполнение всех этапов проекта в правильном порядке.

3.3. Описание реализации

Файл ppo.py состоит из следующих функций:

1. Функция implement_PPO_algorithm (она реализует алгоритм PPO для обучения агента в среде обучения):

- Инициализация среды: если переменная intermediate равна None, вызывается метод set_force_reset у среды env для начальной настройки и затем окружение оборачивается в DummyVecEnv для использования в stable_baselines3.
- Задание гиперпараметров: из hyperparameters извлекаются значения гиперпараметров: имя политики, коэффициент дисконтирования (gamma), коэффициент энтропии (ent_coef), скорость обучения (learning_rate) и количество проходов (num_forward_passes).
- Обучение модели: создается и обучается модель PPO с заданными гиперпараметрами. Модель обучается на основе действий и вознаграждений для определенного количества эпизодов.
- Тестирование модели: после обучения модель тестируется на новых данных (тестовых студентах). В процессе тестирования вычисляется дисконтированный доход для каждого студента, который сохраняется в pred_student_information.

2. Функция save_intermediate_results (она сохраняет промежуточные результаты после каждого прохода обучения):

- Обновление гиперпараметров: количество проходов (num_forward_passes) устанавливается в 1 для начального прохода.
- Цикл обучения: для каждого промежуточного прохода (от 1 до num_forward_passes) вызывается implement_PPO_algorithm с соответствующими параметрами, и результаты сохраняются в указанную директорию.

3. Функция test_individual_env_methods (она предназначена для тестирования различных методов окружения SelectionEnv):

- Тестирование получения наблюдений: Вызывается метод get_observation у окружения для проверки его работы.
- Тестирование обновления способностей: Вызывается метод get_updated_ability.
- Тестирование шага: В цикле вызывается метод step для выполнения шагов в окружении и проверки его работы.

4. Функция create_new_env (она создает новое окружение SelectionEnv с заданными параметрами):

- student_ids - идентификаторы студентов.
- num_test_students - количество тестовых студентов.
- Outputs - выходные данные.
- CONSIDER_TEST_CASES - количество тестовых заданий, которые нужно учитывать.
- MAX_EPISODES - максимальное количество эпизодов.
- Verbose - флаг для вывода дополнительной информации.

5. Функция parse_arguments (она парсит аргументы командной строки с использованием argparse и возвращает их в виде объекта args)

6. Основная функция main (здесь реализуется основной логический поток программы):

- Парсинг аргументов: сначала вызывается parse_arguments для получения аргументов командной строки.
- Создание выходной директории: если директория для сохранения результатов не существует, она создается.
- Чтение данных: с помощью read_dataset читаются данные для студентов и тестовых заданий.

- Создание окружения: вызывается create_new_env для создания нового окружения с заданными параметрами.
- Проверка окружения: проверяется корректность окружения с использованием check_env.
- Гиперпараметрическая оптимизация: если флаг intermediate не установлен, выполняется настройка гиперпараметров, при этом создаются различные комбинации гиперпараметров и для каждой комбинации выполняется обучение модели с использованием implement_PPO_algorithm.
- Сохранение промежуточных результатов: если флаг intermediate установлен, вызывается save_intermediate_results для сохранения промежуточных результатов обучения.

Файл Selection_env.py состоит из следующих методов класса SelectionEnv:

1. Конструктор класса __init__:

- Инициализация параметров происходит посредством конструктора, который принимает параметры, такие как идентификаторы студентов, количество тестовых студентов, выходные данные, количество рассматриваемых тестов и максимальное количество эпизодов. Эти параметры сохраняются как атрибуты класса.
- Загрузка параметров IRT, таких как способность студентов и сложность тестов.
- Загрузка таких данных, как данные о кодах студентов из CSV-файла, словарь кодов для каждого студента, формулировка задания, тестовые случаи.
- Инициализация модели CodeGPT, происходит настройка устройства для вычислений (CPU или GPU), и инициализируются токенизатор и модель CodeGPT для работы с кодом.

- Инициализация переменных отслеживания, происходит установка переменных для отслеживания текущего студента, текущего эпизода, количества эпизодов, тестов для каждого студента и способностей каждого студента.
- Определяются пространства действий и наблюдений для окружения.

2. Метод construct_str_state

- Формируется строковое представление текущего состояния, включающее формулировку задания, код студента и выбранные тесты. Также создается строковое представление для правильного кода.

3. Метод get_observation

- Принимает строковое представление состояния и правильного кода, токенизирует их и пропускает через модель CodeGPT для получения эмбеддингов. Затем эмбеддинги агрегируются, и состояние представляется как разница между эмбеддингами.

4. Метод set_force_reset

- Устанавливает значение отслеживателя студентов и режима работы (обучение или тестирование).

5. Метод reset

- Выполняет сброс окружения, устанавливает начальное состояние, увеличивает значение отслеживателя студентов, если это необходимо, и возвращает начальное наблюдение и информацию.

6. Метод get_updated_ability

- Обновляет способности студента: Метод обновляет способность студента с использованием модели IRT, оценивая выполнение студентом выбранных тестов.

7. Метод step

- Выполняет один шаг в окружении, обновляет список выбранных тестов и способность студента, вычисляет награду и формирует новое состояние. Возвращает следующее состояние, награду, флаг завершения эпизода, флаг обрыва и информацию.

8. Метод render

- Выводит текущую информацию об эпизоде, студенте, выбранных тестах и способности студента на консоль.

9. Метод close

- Закрывает окружение и освобождает ресурсы.

10. Метод seed

- Устанавливает сиды для обеспечения воспроизводимости результатов.

3.4. Используемые библиотеки

1. os - работа с файловой системой, создание директорий, проверка существования файлов и директорий.

2. json - чтение и запись JSON-файлов для сохранения и загрузки параметров, результатов и конфигураций.

3. math - выполнение математических операций, таких как вычисление абсолютных значений.

4. argparse - парсинг аргументов командной строки для настройки параметров запуска скрипта.

5. tqdm - отображение прогресса выполнения циклов, особенно при длительных вычислениях или обработке данных.

6. itertools - генерация различных комбинаций и перестановок для перебора гиперпараметров и других итеративных задач.

7. collections.defaultdict - создание словарей с автоматическим созданием значений по умолчанию, что упрощает работу с вложенными структурами данных.

8. stable_baselines3 - реализация алгоритмов обучения с подкреплением. Используются компоненты для проверки среды (check_env), векторизации среды (DummyVecEnv), а также алгоритмы обучения, такие как PPO, DQN, A2C, DDPG, TD3 и SAC.

9. numpy - обработка массивов и выполнение числовых операций.

10. pandas - работа с табличными данными, чтение CSV-файлов и выполнение операций над датафреймами.

11. torch - работа с тензорами и выполнение операций машинного обучения с использованием библиотек PyTorch, включая модели глубокого обучения.

12. gymnasium (gym) - создание среды для обучения с подкреплением. Определение пространства действий и наблюдений.

13. transformers - работа с моделями трансформеров, такими как GPT-2, для обработки естественного языка и кода. Используется токенизатор и модель GPT-2.

14. utils - вспомогательные функции, такие как получение словаря кодов, формулировка задания, загрузка тестовых случаев и получение идентификатора pivot-кода.

15. IRT.implement_irt - чтение набора данных, работа с IRT моделями, получение загрузчиков данных, установка сидов, получение информации о модели и обучение IRT модели.

16. IRT.load_params - загрузка параметров IRT модели, таких как способности студентов и сложность тестов.

3.5. Результаты

На вход подаются файлы:

- CodeStates.csv -–хранит коды, написанные студентами;
- correct_outputs.json – хранит в формате json правильные ответы на каждый из тестов;
- test_cases.json – хранит тесты на определенную задачу;
- problem.txt – записана поставленная задача;
- all_code_outputs.json – формирует ответы на тесты, выводимые кодом студента, состоит из 0 и 1 для каждого теста по каждому студенту.

На выходе выводятся файлы:

0.900000_0.010000_0.000100_10.json (например) – где через _ написаны параметры: гамма, входной коэффициент, скорость обучения, количество прямых проходов. В этом и подобных файлах содержатся данные о промежуточной оценке навыков студента и тесты, которые могли бы улучшить его код, если он их проанализирует.

Этот и другие файлы с разными параметрами генерируются по пути results/PPO/MlpPolicy.

В коде программы был протестирован следующий пример:

Текст задачи - При наличии массива int возвращает сумму чисел в массиве, за исключением случаев, когда игнорируются разделы чисел, начинающиеся с 6 и продолжающиеся до следующих 7 (за каждым числом 6 будет как минимум одно число 7). Возвращает 0, если чисел нет.

Данные входных и выходных данных можно просмотреть в приложении А.

Всего в тестировании приняло участие 10 студентов. Были сформированы файлы json с кодом каждого студента, который решал поставленную задачу

(CodeStates.csv). Также были переданы файлы correct_outputs.json, test_cases.json, all_code_outputs.json.

По итогам обучения программа выводит файлы с оценками и предлагаемыми тестами и зачастую корректно отрабатывает – ставит положительные оценки тем студентам, у которых код правильно решал поставленную задачу и отрицательные тем, у кого неправильно. Но также можно заметить, что с изменением параметров данные выходных файлов могут меняться. В целом же программа решает поставленную ей задачу.

3.6. Пользовательский интерфейс приложения

В Android Studio был сделан шаблон мобильного приложения для просмотра и размещения обучающих курсов.

Всего было разработано три Activity:

- Регистрация;
- Авторизация;
- Просмотр курсов.

Была использована база данных SQLite для добавления пользователя, создания и обновления таблицы пользователей, получения данных из таблицы. Также была реализована проверка на то, зарегистрирован ли уже пользователь и переход между Activity.

Просмотреть интерфейс мобильного приложения можно в приложении В.

3.7. Постановка задачи на следующий семестр

Задачи на улучшение модели обучения с подкреплением:

- Протестировать модель на большем количестве данных, порядка 100 студентов и больше;
- Использовать другие алгоритмы обучения с подкреплением, например, A2C;
- Добавить в модель возможность анализа односложных ответов студента.

Задачи на улучшение мобильного приложения:

- Добавить возможность выбора роли (преподаватель или студент) на странице регистрации и авторизации;
- В зависимости от роли прописать функционал для преподавателя – возможность создать курс, описать задачу, тестовые данные или вопросы, правильные ответы, ранжирование оценки; и для студента – просмотр и регистрация на курсе, прохождение тестов курса, получение оценки и улучшение навыков в зависимости от оценки;
- Улучшить хранение информации в базе данных.

Общие задачи:

Объединить работу модели и приложения – чтобы из базы данных выгружались json-файлы с нужными данными и после отработки модели данные загружались обратно в базу данных;

Написать сервер, чтобы можно было в режиме реального времени тестировать приложение.

4. Выводы

В рамках проекта была создана система автоматической оценки знаний для адаптивного обучения, направленная на персонализацию образовательного процесса и повышение его эффективности.

Ключевые характеристики проведенных исследований и решенных задач:

- Разработана методика оценки текущего уровня знаний на основе ответов на предложенные вопросы.
- Оценка вероятности правильного ответа учащегося на каждое задание с учетом его текущих знаний и характеристик задания.
- Разработан и поддерживается набор тестовых заданий, из которых система выбирает наиболее подходящие для каждого учащегося.
- Реализован алгоритм выбора наиболее информативных заданий для каждого учащегося, что способствует целенаправленному улучшению знаний.

Ключевые особенности математического обеспечения:

- Модель Тестовых Ответов (IRT) - использована для анализа характеристик тестовых заданий и базовых качеств знаний учащихся. Она позволяет точно оценить, какие задания являются наиболее подходящими для определения уровня знаний учащегося.
- Марковские Процессы Принятия Решений (MDP) - применены для нахождения оптимальной политики выбора заданий, которая максимизирует ожидаемое совокупное вознаграждение (улучшение знаний учащегося) с течением времени.
- Методы динамического программирования, такие как итерация значений и итерация политики, использованы для итеративного обновления оценок значений каждого состояния.

Библиотеки машинного обучения и глубокого обучения:

- Использованы библиотеки, такие как torch и transformers, для создания моделей и обработки данных.
- Применены алгоритмы обучения с подкреплением (например, PPO) для оптимизации процесса обучения.

Средства разработки и тестирования:

- Интеграция с Gymnasium для создания и тестирования окружения обучения с подкреплением.
- Применение средств автоматизации и анализа данных, таких как pandas и numpy, для работы с данными и моделями.

Основные цели проекта были достигнуты в полной мере:

- Разработана система автоматической оценки знаний.
- Создана база тестовых заданий и алгоритм их адаптивного выбора.
- Реализованы и протестированы модели на небольшом количестве данных.

Применяемые методы и средства информационных технологий:

- Продолжение использования методов машинного обучения и алгоритмов обучения с подкреплением.
- Использование средств разработки мобильных приложений и серверной части для интеграции модели и базы данных.

За время выполнения проекта были приобретены следующие компетенции:

- Знания и навыки в области адаптивного обучения и автоматической оценки знаний.
- Опыт работы с моделями IRT и MDP.
- Навыки разработки и тестирования моделей машинного обучения.

- Опыт интеграции моделей с мобильными приложениями и базами данных.

Проект был выполнен успешно, основные цели достигнуты. В следующем семестре планируется расширение и улучшение модели обучения с подкреплением, а также разработка и интеграция функционала для мобильного приложения, что позволит улучшить адаптивность и персонализацию образовательного процесса.

5. Список используемых источников

1. Computerized Adaptive Testing (CAT) // Assessment Systems URL: https://assess.com/computerized-adaptive-testing/.

2. Markov Decision Processes // 21 Lycee Wiki URL: https://21lycee.fandom.com/wiki/Markov_Decision_Processes.

3. Proximal Policy Optimization // OpenAI URL: https://openai.com/index/openai-baselines-ppo/.

4. Stable-Baselines3 Docs // Stable-Baselines3 URL: https://stable-baselines3.readthedocs.io/en/master/.

5. Morten Aa. Petersen, Hugo Vachon, Johannes M. Giesinger, Mogens Groenvold Development of standard computerised adaptive test (CAT) settings for the EORTC CAT Core // Quality of Life Research . - Brussels: the European Organisation for Research, Treatment of Cancer (EORTC) , 2023. - С. 952-955. URL: https://link.springer.com/article/10.1007/s11136-023-03576-x#citeas.

6. Caroline B. Terwee Common measures or common metrics? the value of IRT-based common metrics // Quality of Life Research . - Amsterdam: Springer Link , 2023. - С. 1-4. URL: https://link.springer.com/article/10.1186/s41687-023-00657-w.

7. Царев Р.Ю. Тынченко С.В. Гриценко С.Н. АДАПТИВНОЕ ОБУЧЕНИЕ С ИСПОЛЬЗОВАНИЕМ РЕСУРСОВ ИНФОРМАЦИОННО-ОБРАЗОВАТЕЛЬНОЙ СРЕДЫ // Современные проблемы науки и образования. - Красноярск: Сетевое издание, 2016. - С. 1-3. URL: https://science-education.ru/ru/article/view?id=25227.

8. Адаптивное обучение в высшем образовании: за и против / К. А. Вилкова, Д. В. Лебедев; Национальный исследовательский университет «Высшая школа экономики», Институт образования. — М.: НИУ ВШЭ, 2020 — 36 с. — 200 экз. — (Современная аналитика образования. № 7 (37)).

9. Arnd Hartmanns, Sebastian Junges, Tim Quatmann & Maximilian Weininger A Practitioner’s Guide to MDP Model Checking Algorithms // Lecture Notes

in Computer Science. - Berlin: Springer Link, 2023. - С. 469–488. URL: https://link.springer.com/chapter/10.1007/978-3-031-30823-9_24.

10. Rajmeet Singh Bhourji, Saeed Mozaffari & Shahpour Alirezaee Reinforcement Learning DDPG–PPO Agent-Based Control System for Rotary Inverted Pendulum // Arabian Journal for Science and Engineering. - London: Springer Link, 2023. - С. 1683–1696. URL: https://link.springer.com/article/10.1007/s13369-023-07934-2.

6. Дополнительные разделы отчета

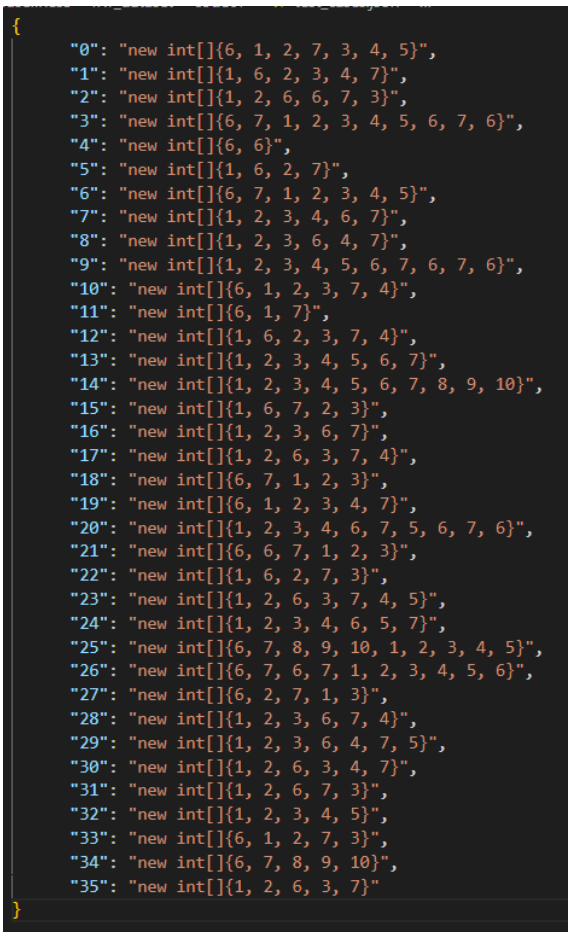
Приложения А

```
CodeStateID,Code
student_1,"public int sum67(int[] nums)
{
    int sum = 0;
    boolean sixMode = false;
    for(int i = 0; i < nums.length; i++)
    {
        if(sixMode)
        {
            if(nums[i] == 7)
                sixMode = false;
        }
        else if(nums[i] == 6)
            sixMode = true;
        else
            sum += nums[i];
    }
    return sum;
}
"
```

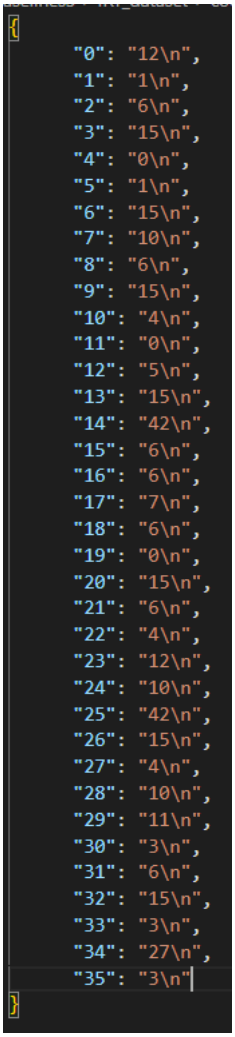
А.1 – пример кода студента 1 в файле CodeStates.csv

```
student_3,"public int sum67(int[] nums)
{
    int sum = 0;
    boolean cont = true;
    for (int num : nums) {
        if (cont) {
            if (num != 6) {
                sum += num;
            }
            else {
                cont = false;
            }
        }
        else {
            if (num == 7) {
                cont = true;
                sum += num;
            }
        }
    }
    return sum;
}
"
```

А. 2 – пример кода студента 3 в файле CodeStates.csv

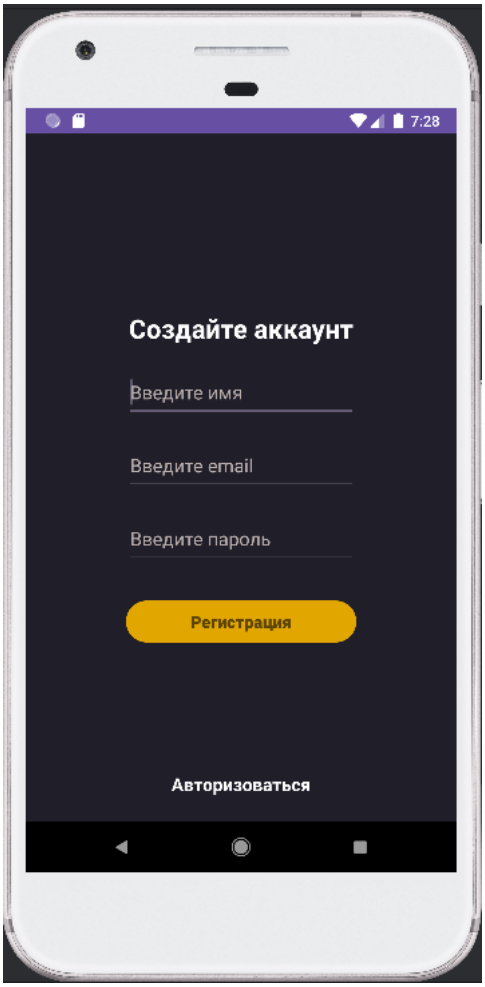


А.3 – Тесты из файла test_cases.json

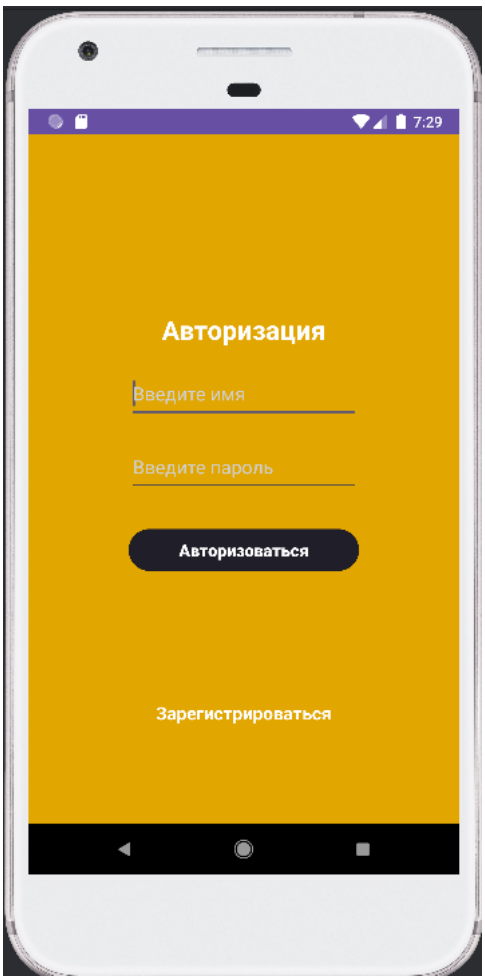


А.4 – Правильные ответы из файла correct_outputs.json

Приложения В



В.1 – страница регистрации



В.2 – страница авторизации

В.1 – страница просмотра курсов