|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация

КАФЕДРА Системы автоматизированного проектирования

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

***Исследование методов анализа параданных при\_\_\_\_\_ прохождении онлайн‑тестов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Студент РК6-41М **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Прокопенко А.П.

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  Берчун Ю.В.

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**               Грошев С.В.

(Подпись, дата) (И.О.Фамилия)

*2021 г.*

РЕФЕРАТ

Расчетно-пояснительная записка к выпускной квалификационной работе магистра «Исследование методов анализа параданных, полученных при прохождении онлайн-тестов» содержит 79 страниц машинописного текста, 22 рисунка, 3 таблицы.

Целью магистерской квалификационной работы является исследование методов анализа параданных, полученных при прохождении онлайн-тестирования.

В данной работе рассмотрены вопросы, связанные со сбором, обработкой и анализом параданных. Также проанализированы различные алгоритмы машинного обучения, позволяющие производить классификацию и кластеризацию параданных. В процессе данного исследования было разработано веб-приложение, отслеживающее действия респондента на странице с онлайн-тестом и сохраняющее на сервере все ключевые события. Полученные данные были обработаны и использованы для задач кластеризации и классификации. В теоретической части проведен анализ различных методов классификации и кластеризации, исходя из результатов вычислительного эксперимента, были сделаны выводы о эффективности данных методов.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 9](#_Toc75911037)

[1. НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ 11](#_Toc75911038)

[1.1. Академическое мошенничество в образовательных процессах 11](#_Toc75911039)

[1.2. Прокторинг в дистанционном образовании 12](#_Toc75911040)

[1.3. Параданные 13](#_Toc75911041)

[1.4. Типы параданных 15](#_Toc75911042)

[1.5. Машинное обучение 18](#_Toc75911043)

[1.6. Выбор задач машинного обучения 19](#_Toc75911044)

[1.7. Использование метрики для оценки эффективности алгоритма 20](#_Toc75911045)

[1.8. Анализ функционала веб-приложений для cоздания тестов и прокторинга 21](#_Toc75911046)

[2. КОНСТРУКТОРСКАЯ ЧАСТЬ 25](#_Toc75911047)

[2.1. Программная реализация 25](#_Toc75911048)

[2.2. Функционал разработанного приложения 25](#_Toc75911049)

[2.3. Отслеживание и запись событий при прохождении теста 27](#_Toc75911050)

[2.4. Первоначальный анализ параданных 28](#_Toc75911051)

[2.5. Анализ тепловой карты взаимодействия пользователя с тестом 34](#_Toc75911052)

[2.6. Построение модели и входные данные 35](#_Toc75911053)

[2.7. Обработка приложением событий во время прохождения теста 37](#_Toc75911054)

[2.8. Результаты классификации наивным Байесовским классификатором 38](#_Toc75911055)

[2.9. Результаты классификации с помощью дерева решений 39](#_Toc75911056)

[2.10. Результаты классификации с применением Логистической регрессии 40](#_Toc75911057)

[2.11. Анализ полученных результатов кластеризации 40](#_Toc75911058)

[3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 44](#_Toc75911059)

[3.1. Общая информация о проведенном тестировании 44](#_Toc75911060)

[3.2. Использование Google Colaboratory 45](#_Toc75911061)

[3.3. Библиотека Pandas 46](#_Toc75911062)

[3.4. Seaborn для визуализации данных 46](#_Toc75911063)

[3.5. Scikit-learn для машинного обучения 47](#_Toc75911064)

[ЗАКЛЮЧЕНИЕ 48](#_Toc75911065)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 49](#_Toc75911066)

[ПРИЛОЖЕНИЕ А 52](#_Toc75911067)

# ВВЕДЕНИЕ

Любой образовательный процесс не представляется возможным без оценки знаний обучающегося с помощью различных контрольных мероприятий. Развитие подходов к оценке результатов тестирования позволяют преподавателю не только оценить знания конкретного студента, но и выяснить, с изучением каких разделов дисциплины у студентов возникли трудности. К примеру, это может выражаться в неполных или неправильных ответах студентов на вопросы раздела этой дисциплины, с которым у студентов возникли трудности.

Академическое мошенничество явление не новое и распространено в различных образовательных учреждениях. Учащиеся используют нечестные способы как при выполнении домашних заданий, так и на экзаменах. За последние полтора года в российских ВУЗах в связи с эпидемиологической ситуацией повсеместно стало использоваться дистанционное обучение.

Обучение, сдача домашних заданий, экзамены, а также защита выпускных квалификационных работ проводятся с использованием различного программного обеспечения, обеспечивающего голосовую, видео и текстовую связь учащихся с преподавателем. Рост распространенности дистанционного обучения дает нечестным студентам дополнительные возможности, по причине того, что не всегда преподаватель может контролировать что у студента отображается на экране устройства, студенты не всегда имеют веб-камеры на своих устройствах, а также нет возможности увидеть, что происходит слева и справа за экраном монитора.

Для решения проблемы контроля честности испытуемого используется прокторинг - процедура наблюдения и контроля за дистанционным испытанием. Эта процедура активно используется при повышении квалификации сотрудников и при приеме на работу, также применим он и в образовательных процессах. Прокторинг включает в себя различные методы контроля честности, но не все из них легко применимы, к примеру, отслеживание направления взгляда или проверка паспортных данных, так как некоторые требуют больших вычислительных мощностей или наличия дополнительного оборудования.

Используя некоторые из методов прокторинга при проведении тестирования учащихся, можно получить определенный набор параданных, представляющих собой информацию о взаимодействии пользователя со страницей в браузере в процессе прохождения тестирования. Применив к этой информации методы обработки данных, получим набор входных данных для различных алгоритмов машинного обучения, результат применения которых представляет вспомогательную метрику, позволяющую преподавателю оценить честность и полноту знаний студента, сравнив полученные данные с классической системой оценки.

# НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

## Академическое мошенничество в образовательных процессах

Проблема академического мошенничества в зарубежных и российских вузах не является новой и в последние десятилетия его распространенность только возрастает. Под академическим мошенничеством подразумевают нечестные приемы, нарушающие правила сдачи контрольных мероприятий и экзамен, которыми пользуются обучающиеся для достижения преимущества в учебе или попытки скрыть отсутствие знаний. Примерами академического мошенничества служат: списывание на контрольных мероприятиях, копирование чужих работ, сотрудничество с другими студентами для выполнения индивидуального задания. Укоренение данных приемов в образовательной среде ведет к снижению уровня квалификации выпускников, снижению эффективности и качества системы образования [2].

Масштабы нечестного поведения обучающихся существенно варьируется в зависимости от вуза. Согласно проведенным исследованиям, доля учащихся, никогда не прибегавших к плагиату, в одном из вузов составляет 29%, и она вдвое выше в другом - 64%. Вариации в масштабах списывания относительно меньше - от 36 до 62% [4]. Студенты высоко оценивают распространенность списывания и плагиата среди своих одногруппников: каждый третий учащийся считает, что большинство одногруппников прибегают к списыванию. Несмотря на то, что большинство студентов считают, что списывание широко распространено среди их одногруппников, две трети учащихся отмечают высокую вероятность строгого наказания за плагиат и списывание, но при этом только половина студентов высоко оценивают вероятность проверки преподавателем работы на наличие плагиата [3].

С увеличением распространенности и повсеместности дистанционного образования в российских вузах, проблема академического списывания не исчезла, поскольку при дистанционной форме обучения, сдача контрольных мероприятий также происходит удаленно. В связи с этим студенты имеют больше возможностей для использования нечестных способов сдачи контрольных мероприятий, использование которых сложнее проконтролировать без дополнительного программного обеспечения или дополнительного оборудования. Выявление нежелательного поведения студентов — одно из применений анализа образовательных данных, полученных в процессе обучения студента [1]. Регулярное использование запрещенных методов способствует формированию привычки уходить от ответственности при решении относительно сложных задач, искать наиболее легкие и не всегда корректные способы достижения целей. Одним из способов решения проблемы контроля использования студентами академического мошенничества может быть сбор параданных и их дальнейший анализ.

## Прокторинг в дистанционном образовании

Меры контроля честности актуальны при обучении сотрудников компании, при проверке кандидатов на должность в компании, а также при тестировании учащихся. Прокторинг — это процедура контроля за ходом дистанционного испытания (в английском языке «proctor» — это наблюдатель на экзаменах в вузе). В связи с вынужденной обязательностью дистанционного обучения появилась необходимость создания сервисов онлайн-прокторинга, при которых наблюдатель (проктор) идентифицирует личность абитуриента или студента, разъясняет правила экзамена, оценивает поведение сдающего и следит за тем, чтобы человек не совершал нарушений.

Системы прокторинга разделяют на три вида:

* пассивный мониторинг программного обеспечения на компьютерах экзаменуемых;
* активное ограничение ПО на компьютерах учеников;
* пассивное видеонаблюдение с помощью ПО, которое обращается к веб-камере студентов для непосредственной фиксации всех действий.
* активное видеонаблюдение. Реализуется аналогично пассивному, но с добавлением мониторинга в режиме реального времени. Один проктор (наблюдатель) может следить за несколькими учащимися одновременно.

Системы прокторинга на рынке представлены как встраиваемыми в LMS (Learning management system — Cистема управления учебным процессом) решениями, так и предоставляющими свой браузер, либо свою LMS для проведения тестирования. В системах прокторинга можно выделить следующие способы контроля при проведении тестирования:

* запись экрана компьютера пользователя;
* запись изображения с веб-камеры;
* запись изображения с веб-камеры дополнительного устройства, находящегося недалеко от пользователя;
* отслеживание взгляда пользователя;
* отслеживание наличия посторонних людей в кадре;
* отслеживание звуков на наличие речи;
* отслеживание попыток параллельной авторизации с других устройств.

## Параданные

В последние три десятилетия в сфере методологии и методов сбора данных широкое распространение получили компьютеризированные и онлайн - способы сбора данных. В настоящий момент многие исследования используют стратегии смешивания методов сбора данных (mixmode), одним из которых зачастую является либо онлайн - опрос, либо опрос с применением компьютерных технологий (телефонный, с использованием планшета). Кроме того, доказанная эффективность подобных способов сбора данных привела к тому, что в некоторых странах компьютеризированные или онлайн - опросы проводятся (или планируются к проведению) как альтернатива личному интервьюированию с бумажной анкетой в случае переписи населения.

Зачастую наиболее важными причинами задействования таких методов сбора данных являются:

1. уменьшение стоимости (особенно в долгосрочной перспективе);
2. снижение затрат на логистику (по сравнению с личными интервью с бумажными анкетами) - печать анкет, перевозка, ввод данных;
3. сокращение времени, требуемого для перевода имеющихся анкет в формат базы данных.

Кроме того, очень важным преимуществом стоит считать возможность сбора дополнительных данных, описывающих разные этапы процесса сбора данных (как сторону интервьюера, так и сторону респондента) - параданные.

Термин «параданные» был впервые определен М. Купером в 1998 году как побочные данные о процессе сбора данных, получаемые в результате использования компьютеризированных методов сбора данных. Важно отметить, что методологические исследования, в рамках которых использовались параданные, существовали и до этого (например, сведения о результате установления связи с респондентом употреблялись для определения более эффективного способа отбора респондентов).

Тем не менее, распространение применения подобного типа данных и более комплексное их осмысление наблюдаются именно в последние два десятилетия, что связывается с широким распространением способов сбора данных, основанных на использовании компьютерных или онлайн — технологий, а также с повышением интереса к анализу процесса сбора данных, выявлению ошибок и изучению возможностей их сокращения.

Исследования с применением параданных показывают их состоятельность в случае изучения различных аспектов методологии исследований, таких как описание эффектов интервьюера, расчет корректировочных коэффициентов с учетом уровня неответов - отказов от ответов, контроль процесса отбора респондентов интервьюерами, анализ коммуникации между интервьюером и респондентом и опросного инструментария в целом, оценка качества данных и т. д. Проще говоря, параданные дают возможность заглянуть в процесс сбора информации, понять особенности поведения респондентов, интервьюеров, ситуации.

Параданные — это сведения о процессе сбора данных, которые зачастую применяются для оптимизации и повышения эффективности такого процесса.

Зачастую наиболее часто используемые типы параданных (например, время заполнения анкеты, результаты установления связи с респондентом, оценка интервьюером ситуации интервью) не требуют дополнительных финансовых вложений для их сбора и анализа. Это особенно справедливо в случае компьютеризированных и онлайн - опросов, для которых во многих платформах, занимающихся таким методом сбора данных, имеются встроенные инструменты для сбора наиболее общих параданных, которые при этом могут быть сведены к индивидуальному уровню каждого респондента и каждого вопроса. Стоит оговориться, что имеются также типы параданных, которые связаны с более трудоемкими процессами как сбора, так и анализа.

## 1.4. Типы параданных

На текущий момент в сфере исследований методологии социальных наук отсутствует единое определение этого термина. Нередко имеющиеся определения могут даже в - какой-то степени противоречить друг другу.

Наиболее часто упоминается уже приведенное в нашей статье определение М. Купера, приведенное выше и представленное на конференции Американской социологической ассоциации. Развитие исследований с применением параданных на текущий момент указывает, что такое определение упускает значительную часть параданных, использующихся для поправки на ошибку неответов (например, наблюдения и оценки интервьюеров относительно респондента, домохозяйства, района и ситуации интервью в целом).

В этом контексте необходимо выделить наиболее важные характеристики параданных, а также кратко описать их отличие от сходных типов данных: вспомогательных (auxiliary data) и метаданных (metadata). Важно заметить, что нередко исследователи методологии социальных наук пренебрегают обсуждением возможного определения параданных в пользу обсуждения способов их сбора и использования.

Параданные могут быть описаны как дополнительные сведения, замеряемые в процессе заполнения опроса/интервью. Это данные о самом процессе, включающие поведенческие характеристики интервьюера, респондента, описание ситуации прохождения опроса/интервью, в том числе замеряемые с помощью вспомогательных устройств. Параданные недоступны перед проведением полевого этапа, но возникают и меняются в этом процессе, при этом они предоставляют возможность увидеть наиболее значимое поведение респондентов и интервьюеров при сборе данных (проведение интервью, заполнение онлайн - анкеты и т. п.)

Выделяют несколько важных особенностей параданных. Они фиксируются на каждом этапе сбора данных и в подавляющем большинстве случаев являются нереактивными (то есть собираются без ведома или согласия﻿ со стороны респондентов, а также без указания на цели сбора такого рода данных.

Параданные неразрывно связаны с метаданными и вспомогательными данными. Нередко параданные описываются как одна из разновидностей метаданных (поведенческие). Тем не менее, в последние годы зачастую параданные выделяют как отдельную категорию наравне с метаданными.

Метаданные описывают решения относительно различных этапов сбора данных, позволяющие понять структуру имеющихся данных. В отличие от параданных, такие сведения могут быть доступны до начала самого процесса их сбора, не отличаются по респондентам и интервьюерам и позволяют понять, каким образом были собраны данные (построение выборки, инструментарий, методы сбора данных и т. п.), но не дают возможность анализировать поведение интервьюеров или респондентов и ситуацию каждого отдельно взятого интервью.

Вспомогательные данные содержат информацию о респондентах или их группах, позволяющую использовать в анализе сведения, на которые не было получено прямого ответа от респондента в процессе интервью, но которые имеются в распоряжении у исследователей (например, связь результатов опроса с административными базами данных).

Вспомогательные данные отличаются от параданных тем, что так же, как и метаданные, зачастую существуют до начала процесса их сбора; не обязательно описывают такой процесс (нередко это информация из административных источников о характеристиках респондента, интересующих исследователей); перед началом их сбора можно определить, какие типы параданных собирать дополнительно, тогда как для вспомогательных данных приходится ограничиваться тем, что есть изначально.

Обобщая приведенные особенности параданных, стоит отметить, что такой тип сведений описывает поведенческие характеристики респондентов и интервьюеров в процессе интервью или опроса, ситуацию интервью в целом и особенности респондента, домохозяйства или района, в котором проходило интервью, отмечаемые интервьюером.

Выделяются следующие отличительные черты параданных:

- они недоступны перед проведением полевого этапа, но появляются и меняются в процессе сбора данных;

- фиксируются на каждом этапе сбора данных;

- являются нереактивными (не предполагают активного участия респондентов в их сборе и зачастую не указывают отвечающему на то, что подобные сведения собираются и будут каким-то образом использованы в анализе);

- имеют различные уровни детализации и генерализации (например, среднее время заполнения всей анкеты, время заполнения анкеты одним респондентом, среднее время заполнения одного вопроса, время заполнения

отдельно взятого вопроса одним респондентом), в том числе могут описывать индивидуальный уровень одного респондента и отдельного вопроса.

Исключение составляют параданные из предыдущих исследований или волн исследований выбранного респондента.

## Машинное обучение

Термин «машинное обучение» был впервые введён в 1952 году. Его ввел Артур Сэмюель — создатель программы, которая смогла «научиться» играть в шашки лучше, чем её создатель. Таким образом, программа, продемонстрировавшая способность к самообучению на основе своего предыдущего опыта, опровергла суждение о том, что компьютеры способны выполнять только строго заданные для них алгоритмы. Артур Сэмюель определил машинное обучение, как «методы, позволяющие компьютерам учиться без непосредственного их программирования».

Более формальное определение машинного обучения дал американский учёный в области науки о данных Том Митчелл: «Говорят, что компьютерная программа обучается на основе опыта *E* по отношению к некоторому классу задач *T* и меры качества *P*, если качество решения задач из *T*, измеренное на основе *P*, улучшается с приобретением опыта *E*».

Таким образом, машинное обучение представляет собой подраздел искусственного интеллекта, стоящий на стыке таких дисциплин, как математика, статистика, теория вероятностей, теория графов и изучающий алгоритмы, способные самостоятельно обучаться на основе опыта.

Благодаря машинному обучению компьютеры учатся распознавать на фотографиях и рисунках не только лица, но и пейзажи, предметы, текст и цифры. Что касается текста, то и здесь не обойтись без машинного обучения: функция проверки грамматики сейчас присутствует в любом гаджете. Причем учитывается не только написание слов, но и контекст, оттенки смысла и другие тонкие лингвистические аспекты. Кроме того, различные алгоритмы машинного обучения используются в системах прокторинга для распознавания лиц и голоса респондентов при проведении тестирования, а также для регрессионного анализа тестирования на полиграфе [5].

## 1.6. Выбор задач машинного обучения

Большинство задач, решаемых с помощью алгоритмов машинного обучения, можно отнести к одной из следующих категорий:

* задача регрессии - прогноз на основе выборки объектов с различными признаками. К примеру, цена квартиры, стоимость ценной бумаги по прошествии полугода, ожидаемый доход магазина на следующий месяц, качество вина при слепом тестировании;
* задача классификации - получение категориального ответа на основе набора признаков. Имеет конечное количество ответов: является ли изображение человеческим лицом, болен ли пациент раком;
* задача кластеризации - распределение данных на группы: разделение всех клиентов мобильного оператора по уровню платёжеспособности, отнесение космических объектов к той или иной категории;
* задача уменьшения размерности - сведение большого числа признаков к меньшему (обычно 2–3) для удобства их последующей визуализации (например, сжатие данных);
* задача выявления аномалий - отделение аномалий от стандартных случаев. На первый взгляд она совпадает с задачей классификации, но есть одно существенное отличие: аномалии – явление редкое, и обучающих примеров, на которых можно натренировать обучающуюся модель на выявление таких объектов, либо исчезающе мало, либо просто нет, поэтому методы классификации здесь не работают. На практике такой задачей может являться выявление мошеннических действий с банковскими картами.

Применительно к данной работе, накопленные в процессе прохождения онлайн-теста студентами данные будут использованы для анализа, а также решения задач классификации и кластеризации. При классификации полученный набор данных будет размечен на несколько классов, каждый класс будет соответствовать различному режиму прохождения теста респондентом. Такой выбор обоснован решением исследовать возможность разделения результатов прохождения теста на классы в зависимости от честности и уровня знаний респондента.

Основными параметрами для формирования классов в задаче классификации выступают время просмотра студентом каждого задания теста и взаимодействие студента с веб-страницей, а именно действия по изменению текущей вкладки, разрешения экрана и т. д. Подробнее процесс разметки данных рассмотрен в конструкторской части.

## 1.7. Использование метрики для оценки эффективности алгоритма

Существует множество различных метрик для оценки эффективности используемого алгоритма машинного обучения. Рассмотрим самые популярные из них и объясним итоговый выбор метрики.

Самой простой является метрика accuracy – показатель, описывающий общую точность предсказания модели по всем классам. Данная метрика вычисляется по следующей формуле:

где TP, TN, FP и FP являются элементами матрицы несоответствий. Данная матрица имеет следующий вид:

Таблица 1. Матрица несоответствий

|  |  |  |
| --- | --- | --- |
|  | y = 1 | y = 0 |
| ŷ | True Positive (TP) | False Positive (FP) |
| ŷ | False Negative (FN) | True Negative (TN) |

TP – true positive, классификатор верно отнес объект к рассматриваемому классу.

TN – true negative, классификатор верно утверждает, что объект не принадлежит к рассматриваемому классу.

FP – false positive, классификатор неверно отнёс объект к рассматриваемому классу.

FN – false negative, классификатор неверное утверждает, что объект не принадлежит к рассматриваемому классу.

Следующей рассмотрим метрики precision и recall.

Precision – точность, показывает долю объектов класса, среди объектов выделенных классификатором.

Recall – полнота, показывает долю найденных объектов класса к общему числу объектов класса.

Последней рассмотрим одну из самых популярных и достаточно эффективных метрик – F-меру.

Метрика измеряет эффективность классификатора, учитывая recall в β раз важнее, чем precision. Мы будем использовать часто используемое значение β равное 1, позволяющее одинаково учитывать важность значений precision и recall.

## 1.8. Анализ функционала веб-приложений для cоздания тестов и прокторинга

Существует множество различных web-сервисов, предоставляющих различный функционал для создания и проведения тестирования онлайн, прокторинга во время контрольных мероприятий. Был проведен сравнительный обзор и краткое ознакомление с существующими платными и бесплатными российскими и зарубежными решениями. Большинство сервисов для создания тестов предоставляют следующий функционал для создания тестов:

* выбор одного правильного варианта
* выбор нескольких правильных вариантов
* ввод текстового ответа
* выбор соответствия нескольких вариантов друг
* загрузка изображения, аудиофайла.

В качестве системы оценки подавляющее большинство web-сервисов предлагают следующее:

* указание числа баллов за каждый правильный ответ
* начисление штрафных баллов за неправильный ответ

Статистика по прохождению созданных тестов чаще всего содержит время прохождения теста и процент правильных ответов. Дополнительные метрики сервисы либо не предоставляют, либо требуют наличия подписки.

Для применения прокторинга при проведении контрольных мероприятий также существует множество различных сервисов. Большинство сервисов предоставляют следующие возможности:

* отслеживание движений головы;
* отслеживание присутствия других людей в кадре;
* отслеживание наличия разговора во время тестирования;
* видеозапись тестирования;
* запись экрана компьютера.

Помимо указанных, общих практически для всех сервисов возможностей, были найдены также достаточно редкие возможности:

* прохождение тестирования в защищенном браузере;
* анализ эмоционального отклика;
* уведомления о событиях во время тестирования;
* дополнительная видеофиксация с мобильного устройства, установленного сбоку от проходящего тест пользователя.

Анализ существующих web-сервисов показал, что функционал бесплатных сервисов для проведения тестирования, использования прокторинга и оценки результатов является недостаточным в ключе использования гибкой системы формирования и оценки тестовых заданий.

Если же говорить о методах прокторинга, то большинство сервисов предоставляют необходимый функционал по платной подписке, к тому же использование веб-камер при прохождении теста всеми испытуемыми достаточно сложно реализовать в рамках прохождения теста студентами.

В связи с этим, рассмотренные сервисы по тем или иным причинам не подходят для поставленной в рамках данной работы задачи. Было принято решение самостоятельно разработать web-приложение, имеющее следующий функционал:

* выбор нескольких/ всех правильных вариантов ответа;
* выбор одного правильного ответа;
* хранение и редактирование различных наборов тестов;
* указание онтологических данных для каждого теста;
* сохранение времени прохождения теста;
* запись активности пользователя во время прохождения теста;
* единовременное отображение одного вопроса на странице;
* запрет на копирование текста и использование некоторых сочетаний клавиш;
* сохранение очередности выбора и отмены выбора ответов;
* оценка результатов тестирования на основе баллов за каждый из ответов;
* оценка результатов тестирования в виде объема знаний понятийной базы;
* оценка результатов тестирования с использованием метрик, направленных на сбор слабоформализованной информации.

# КОНСТРУКТОРСКАЯ ЧАСТЬ

## Программная реализация

Разработанная система является клиент-серверным приложением. Клиентская часть исполняется веб-браузером, а серверная часть приложения развёрнута на облачном сервере, там же расположена и база данных. Вся логика работы приложения завязана на взаимодействии различных сущностей между собой.

Клиентская часть приложения реализована на языке TypeScript, который при подготовке к исполнению в браузере компилируется в JavaScript. Основным фреймворком для вёрстки является React, хранилище клиентских данных представлено фреймворком Flexis-Redux. Доступ к базе и работа с данными осуществляется с помощью СУБД MongoDb.

Клиент представлен двумя контроллерами – Edit и Run – отвечающими, соответственно, за создание/редактирование и прохождение теста. Для доступа к редактированию теста необходимо знать его уникальный ключ, который генерируется при создании.

## Функционал разработанного приложения

Для сбора образовательных данных и их последующего анализа разработанное веб-приложение должно имеет в себе функционал, позволяющий отслеживать и записывать следующие данные:

* время начала и окончания прохождения теста;
* все нажатия клавиш мыши во время прохождения теста;
* периодическая запись положения курсора мыши пользователя;
* время просмотра каждого вопроса в тесте;
* текущая активная вкладка браузера;
* отслеживание нажатия специальных комбинаций клавиш;
* изменения в вариантах ответа;
* размер текущего окна браузера;
* разрешение экрана.

Каждое прохождение теста вызывает создание в БД сущности RESULT, содержащей в себе массивы id сущностей ответов и id сущностей событий, записанных в процессе прохождения теста пользователем.

Приложение имеет функционал, позволяющий через web-интерфейс создавать различные наборы тестовых заданий. Доступ к этому функционалу осуществляется с помощью url и специального ключа. Каждое задание не ограничено по числу вопросов, число заданий в тесте также не ограничено. Созданные тестовые задания можно редактировать или удалять. Каждый созданный тестовый набор имеет свой уникальный url и специальный ключ для редактирования. Внешний вид страницы теста в режиме редактирования представлен на рисунке 1.

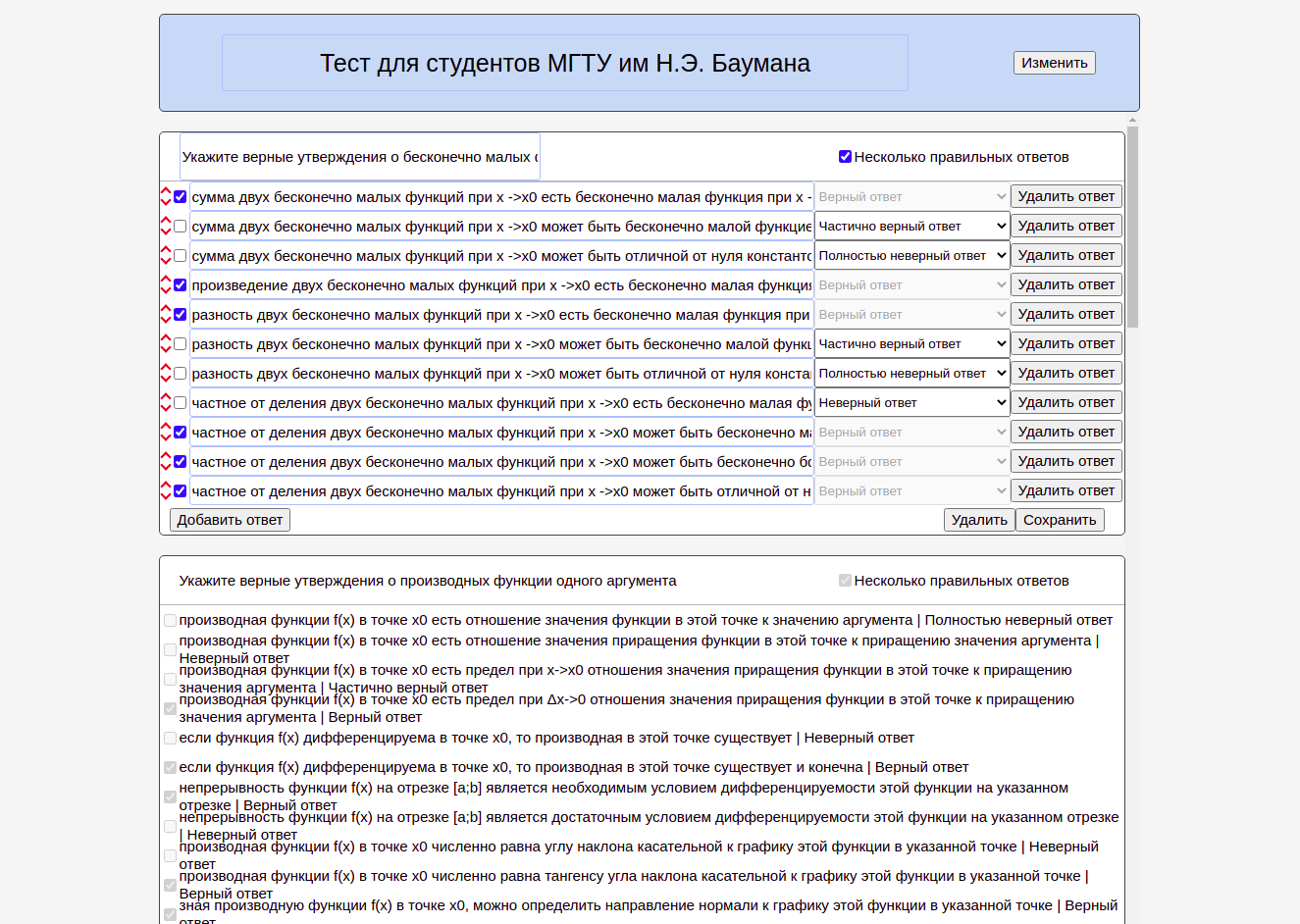


Рис. 1. Страница с редактируемым тестом

При переходе по url без ключа пользователь сможет только пройти тест, без возможности его отредактировать. При прохождении теста после ответа на вопрос правильные ответы не отображаются для пользователей, это сделано для того, чтобы пользователи не отвлекались на проверку своих ответов и подсчет результатов.

Вид страницы во время прохождения теста пользователем представлен на рисунке 2. Во время прохождения теста пользователь имеет возможность переключаться между вопросами, ответ на который он уже дал. Это необходимо для проверки своих ответов перед окончанием теста, если это будет необходимо.

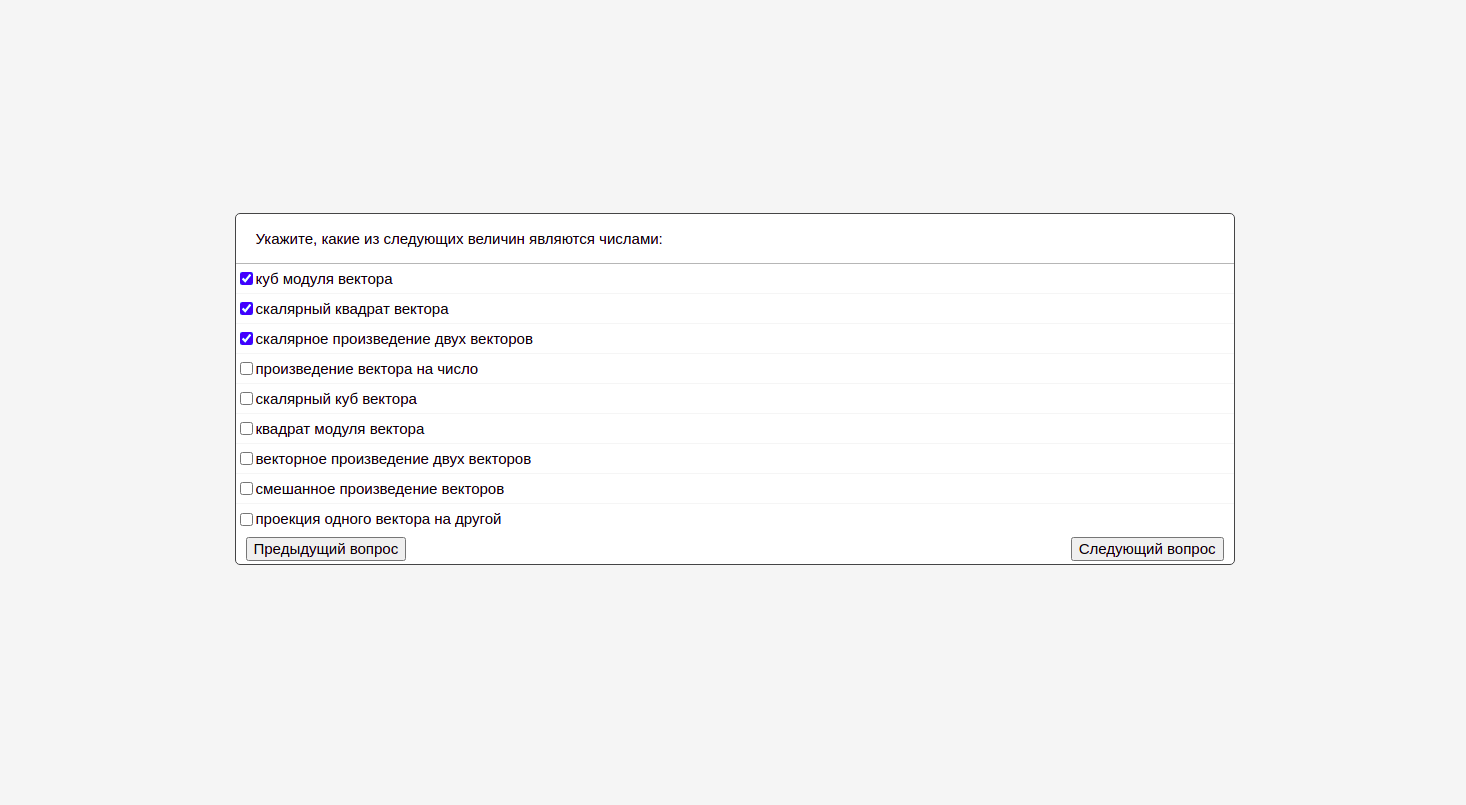


Рис. 2. Страница с вопросом во время прохождения теста

## Отслеживание и запись событий при прохождении теста

Разработанное web-приложение имеет ряд особенностей, которые мы рассмотрим далее. Перед началом прохождения теста пользователь нажимает кнопку “Начать тест” и начинается отсчет время прохождения теста, завершающийся при нажатии на кнопку “Закончить тест”. Нажатие вышеуказанных кнопок вызывает создание в базе двух сущностей с типом TEST\_START и TEST\_FINISH. Стоит заметить, что тип — это название атрибута в базе данных.

В код страницы web-приложения был добавлен запрет на копирование текста на странице. Если пользователь все же захочет скопировать текст на странице воспользовавшись инструментами разработчика, то это событие с типом CONSOLE\_OPENED будет записано в базе данных.

При изменении пользователем размеров открытого окна браузера для того, чтобы воспользоваться какими-то учебными материалами или поиском, в базе данные будет создано соответствующее событие с типом RESIZE. Переключение вкладки браузера или же открытие новой вкладки вызывает создание двух событий с типом FOCUS\_OUT и FOCUS\_IN.

Также записываются координаты положения курсора мыши и нажатия клавиш мыши. Если курсор мыши находится над каким-то элементов интерфейса, к примеру, вариантом ответа на вопрос, то помимо координат курсора мыши записывается и номер варианта ответа. Помимо нажатий клавиш мыши, записываются нажатия на элементы интерфейса - кнопки переключения следующего или предыдущего вопроса, флаговые кнопки вариантов ответа на вопрос.

## Первоначальный анализ параданных

Первым шагом перед выгрузкой данных и их анализом является импорт всех необходимы для работы с данными библиотек. Импорт библиотек представлен на рисунке 3. Сначала необходимо выгрузить из БД накопленные в процессе тестирования данные. Для выгрузки данных из БД была использована библиотека pymongo. После подключения к БД и выбора сущностей для загрузки достаточно только указать параметры фильтрации для выгрузки. Выгрузка данных из БД представлена на рисунке 4.

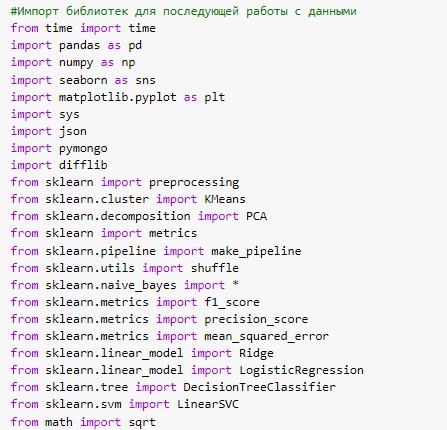


Рис. 3. Импорт библиотек

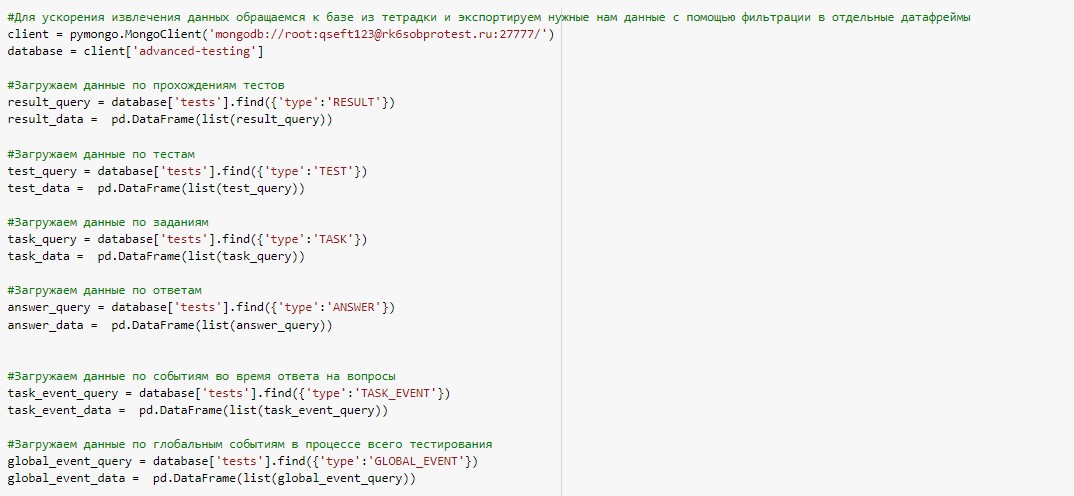


Рис. 4. Извлечение результатов из БД

Поскольку данные в БД представлены различными сущностями с различным набором атрибутов, было решено выгружать данные из БД разделив их на несколько групп. Рассмотрим эти группы подробнее.

Данные сущностей со значением RESULT в колонке type. Заметим, что в дальнейшем сущности будем именовать по значению в колонке type или eventType, если в колонке type значения нет. Сущность RESULT представляет собой уникальную запись об одном прохождении теста одним пользователем. То, как выглядят загруженные данные представлено на рисунке 5.

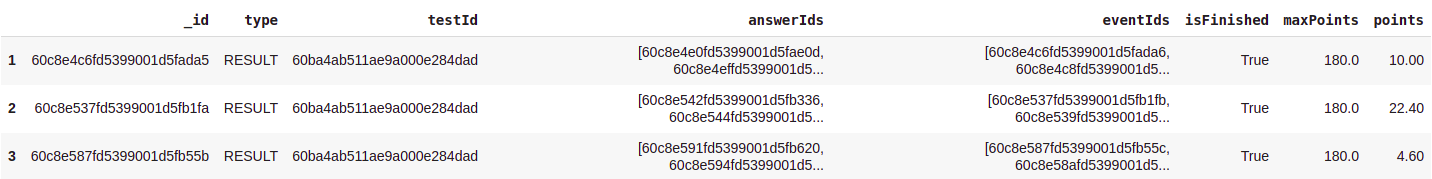


Рис. 5. Загруженные данные сущностей с типом RESULT

Сущность RESULT имеет следующие атрибуты:

* \_id - уникальный идентификатор для записей в БД;
* type - атрибут, указывающий тип сущности;
* testId - атрибут, содержащий в себе уникальный идентификатор теста, который проходил пользователь;
* answerIds - атрибут, содержащий в себе массив с уникальными идентификаторами сущностей ответов пользователя на вопросы теста;
* eventIds - атрибут, содержащий в себе массив с уникальными идентификаторами сущностей событий, записанным во время прохождения пользователем теста;
* isFinished - атрибут, содержащий в себе данные о том, был ли пройден пользователем тест до конца;
* maxPoint - атрибут, содержащий в себе максимальное число баллов за прохождение теста;
* points - атрибут, содержащий в себе чисто баллов, набранное пользователем во время прохождения теста.

На следующем шаге выгружаем данные по сущностям TEST, содержащим в себе данные о вопросах. Вид строки с данными сущности TEST представлен на рисунке 6.

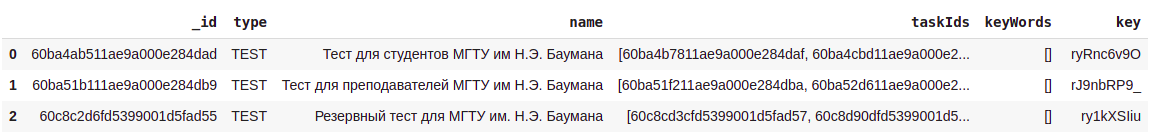


Рис. 6. Загруженные данные сущностей с типом TEST

Рассмотрим атрибуты сущностей TEST:

* \_id - уникальный идентификатор для записей в БД;
* type - атрибут, указывающий тип сущности;
* name
* taskIds
* keyWords
* key

На следующем шаге выгружаем данные по сущностям TASK, содержащим в себе данные о вопросах и вариантах ответа конкретного теста. Вид строки с данными сущности TASK представлен на рисунке 7.

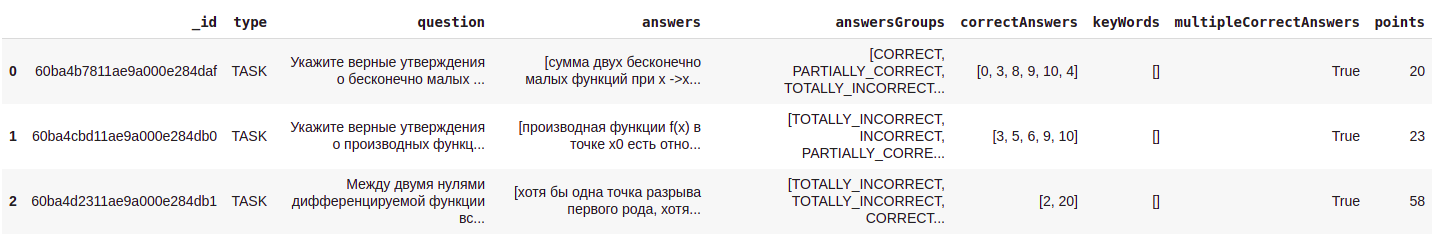


Рис. 7. Загруженные данные сущностей с типом TASK

Рассмотрим атрибуты сущностей TEST:

* \_id - уникальный идентификатор для записей в БД;
* type - атрибут, указывающий тип сущности;
* question
* answers
* answersGroups
* correctAnswers
* keyWords
* multipleCorrectAnswers
* points

На следующем шаге выгружаем данные по сущностям ANSWER, содержащим в себе данные о ответах всех пользователей на все вопросы всех тестов. Вид строки с данными сущности ANSWER представлен на рисунке 8.

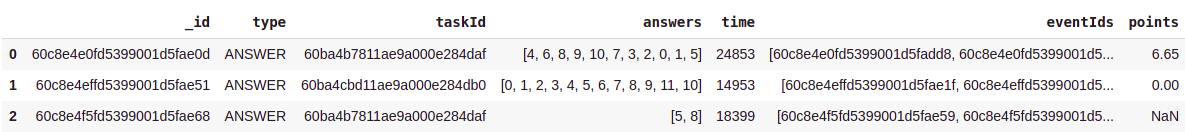


Рис. 8. Загруженные данные сущностей с типом ANSWER

Рассмотрим атрибуты сущностей TEST:

* \_id - уникальный идентификатор для записей в БД;
* type - атрибут, указывающий тип сущности;
* taskId
* answers
* answers
* time
* eventIds
* points

На следующем шаге выгружаем данные по сущностям локальных событий, которые были записаны в процессе взаимодействия пользователя с вопросами и ответами. Набор данных содержит следующие сущности:

* MOUSE\_MOVE
* MOUSE\_CLICK
* CHECKBOX\_CHECKED
* CHECKBOX\_UNCHECKED

Вид строки с данными указанных сущностей представлен на рисунке 9.

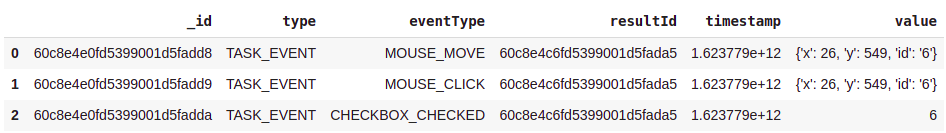


Рис. 9. Загруженные данные локальных событий

Рассмотрим атрибуты сущностей локальных событий:

* \_id - уникальный идентификатор для записей в БД;
* type
* eventType
* resultId
* timestamp
* value

На следующем шаге выгружаем данные по сущностям глобальных событий, которые были записаны в процессе взаимодействия пользователя с тестом и окном браузера. Набор данных содержит следующие сущности:

* RESULT\_CREATED
* TEST\_START
* NEXT\_TASK
* RESIZE
* TEST\_FINISH
* FOCUS\_OUT
* FOCUS\_IN
* PREVIOUS\_TASK
* CONSOLE\_OPENED.

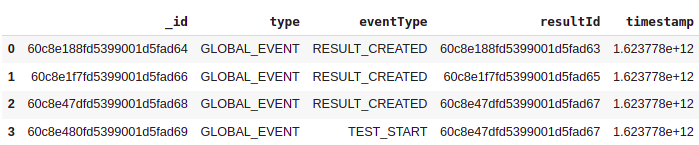
Вид строки с данными указанных сущностей представлен на рисунке 10.

Рис. 10. Загруженные данные глобальных событий.

## Анализ тепловой карты взаимодействия пользователя с тестом

Собранные параданные позволяют нам изучить взаимодействие пользователя с интерфейсом системы для тестирования и построить тепловую карту всех нажатий клавиши мыши и движений курсора мыши пользователя в момент ответа на вопросы.

Поскольку различные вопросы имеют разное число ответов, то тепловые карты будут отличаться в зависимости от конкретного вопроса. Тепловая карта ответа одного из пользователей на 1 вопрос представлена на рисунке 11.

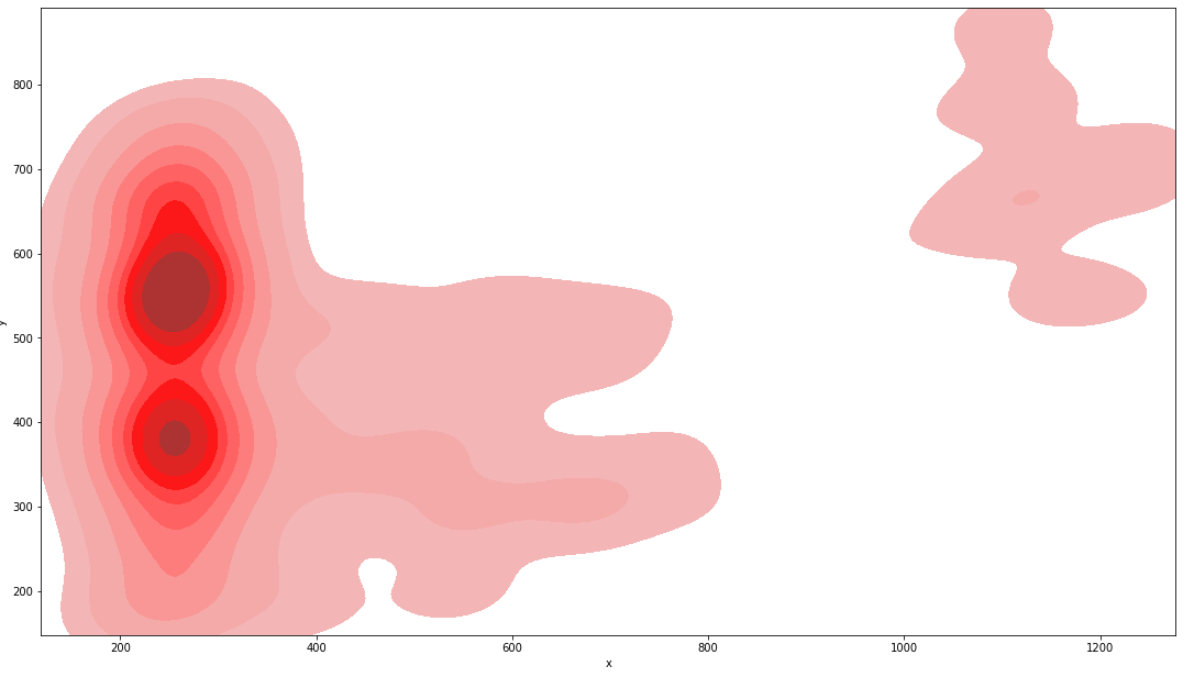


Рис. 11. Тепловая карта взаимодействия пользователя

Эту же карту можно представить в виде точек с координатами курсора мыши. Данное представление отображено на рисунке 12.

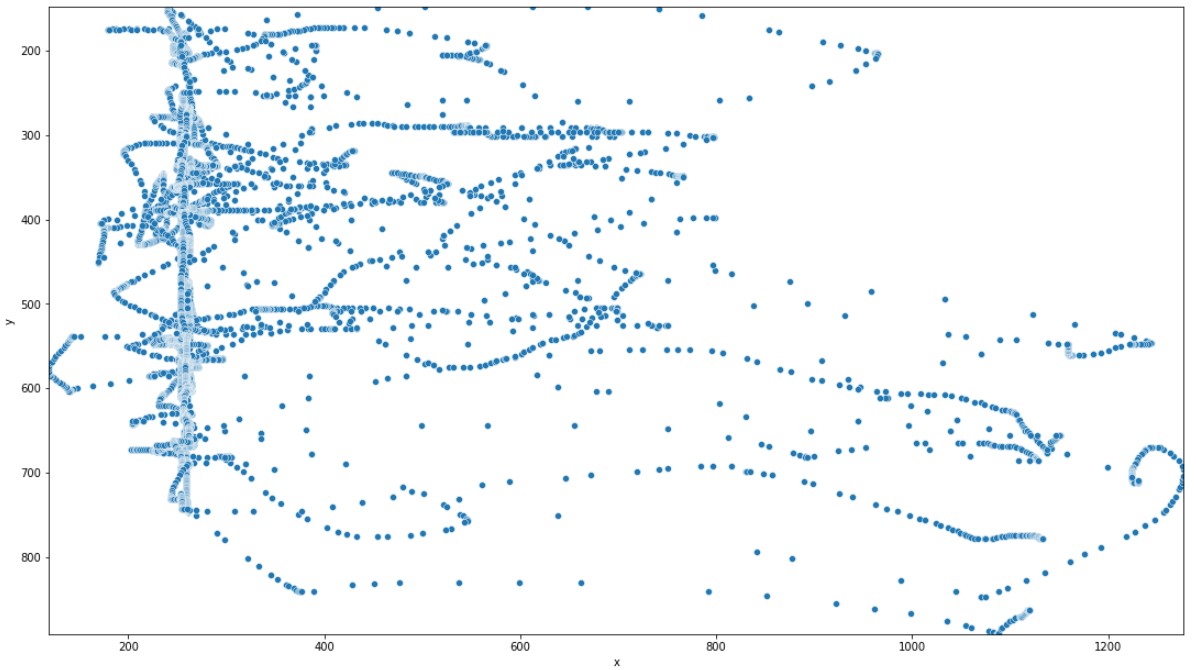


Рис. 12. Карта с координатами точек положения курсора мыши

## Построение модели и входные данные

После первичной обработки полученных параданных необходимо сформировать итоговый набор входных данных для моделей машинного обучения.

Было принято решение выделить следующие признаки для итогового набора данных:

* время просмотра каждого тестового задания;
* число нажатий клавиш мыши в процессе ответа на каждое тестовое задание;
* чисто активации флажковой кнопки в процессе ответа на каждое тестовое задание;
* снятие активации флажковой кнопки в процессе ответа на каждое тестовое задание;
* число событий изменения активной вкладки в процессе ответа на задания теста;
* число событий открытия инструментов разработчика в процессе ответа на задания теста;
* число событий изменения размеров окна браузера в процессе ответа на задания теста.

Для разметки полученных данных была написана функция, которая в зависимости от числа мошеннических действий пользователя и времени просмотра вариантов

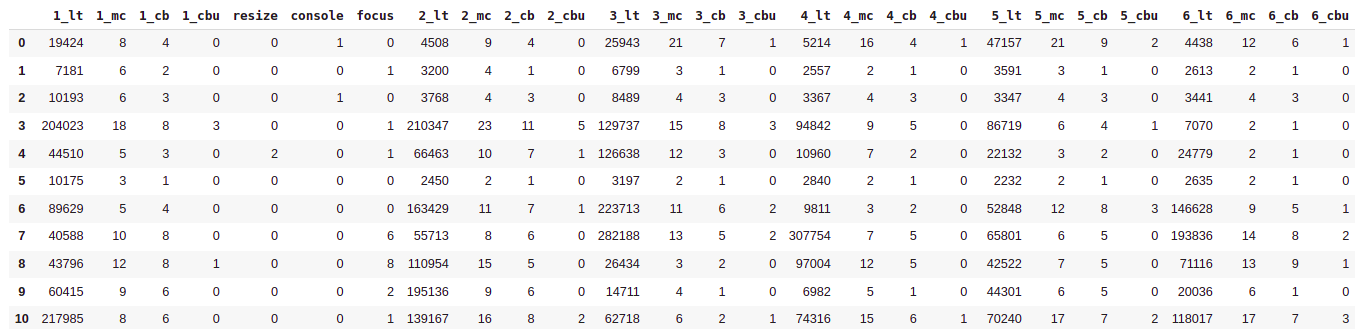
Итоговый набор данных имеет вид, представленный на рисунке 13:

Рис. 13. Итоговый вид параданных после обработки

Сформировав итоговый набор данных посмотрим на данные по всех колонкам датасета. Сделаем это с помощью функции describe. Результат работы функции представлен на рисунке 14.



Рис. 14. Характеристики колонок датасета

Все полученные данные были обработаны в облачном сервисе Colab с использованием различных функций программной библиотеки Pandas.

## Обработка приложением событий во время прохождения теста

Необходимо так же рассмотреть все события и обработчики для их фиксации в момент возникновения такого события. Все отслеживаемые события разделены на два типа – глобальные и локальные. Подробнее эти события описаны в технологической части работы. Сейчас же рассмотрим подробнее обработчики этих событий.

Каждое локальное или глобальное событие заносится в базу данных в виде отдельной сущности. Разница в том, что для оптимизации взаимодействия клиента и сервера локальные события хранятся на клиенте в течении времени ответа на один вопрос теста.

Также при изменении уже данного ответа на вопрос теста и переключении заданий, данные локальных событий отправляются на сервер вместе с измененным ответом на вопрос. Глобальные же события хранятся в течении прохождения всего теста и отправляются на сервер в момент окончания теста.

Для события изменения размеров окна браузер был создан обработчик события resize. Поскольку пользователь физически не может достаточно быстро менять размеры окна браузера был создан декоратор throttle с ограничением в 100 мс.

В настоящее время нет достоверного способа проверить открыты ли на текущей вкладке браузера инструменты разработчика. В связи с этим было принято решение разделить проверку на 2 этапа. Если изменяется горизонтальный размер текущего окна браузера, то такое событие распознается как открытие инструментов разработчика. Аналогично происходит в случае, если соотношение вертикального размера видимой области к вертикальному размеру окна браузера имеет значение менее чем 0.8. Такое соотношение было выбрано из-за того, что часть доступной области окна браузера закрыта адресной строкой и панелями инструментов.

В течение всего времени прохождения теста записываются положения и курсора мыши, а также указывается находился ли в этот момент времени курсор над каким-либо элементом веб-страницы. Для отслеживания событий установки фокуса над элементом и снятия фокуса с элемента были переопределены стандартные функции focus и onblur.

Также в течение всего времени прохождения теста записываются нажатия курсора мыши, а также указывается находился ли в этот момент времени курсор над каким-либо элементом. Для фиксации таких действий были созданы обработчики событий mousemove и onclick. Для события mousemove был дополнительно был создан декоратор throttle с ограничением в 10 мс, поскольку нет необходимости записывать положения курсора мыши с такой частотой.

## Результаты классификации наивным Байесовским классификатором

Наивный байесовский алгоритм (НБА) – это алгоритм классификации, основанный на теореме Байеса с допущением о независимости признаков. Другими словами, НБА предполагает, что наличие какого-либо признака в классе не связано с наличием какого-либо другого признака, т.е. считается, что они вносят независимый вклад. В связи с таким допущением алгоритм называется «наивным».

При классификации полученного набора параданных с помощью НБА был получен следующий результат метрики F1:

## Результаты классификации с помощью дерева решений

Дерево решений – классификатор, построенный на основе решающих правил вида «если, то», упорядоченных в древовидную иерархическую структуру. Разбиение производится с помощью решающих правил, в которых осуществляется проверка значений атрибутов по заданному условию. Структурно дерево решений состоит из объектов двух типов – узлов (node) и листьев (leaf).

В узлах дереве решений указываются правила, разбивающие содержащиеся в нем наблюдения, и производится дальнейшее ветвление. В листьях правил нет, они помечаются меткой класса, объекты которого попали в данный лист. Ветвление в листьях не производится, и они заканчивают собой ветвь дерева (поэтому их иногда называют терминальными узлами). Структура дерева решений представлена на рисунке 15.

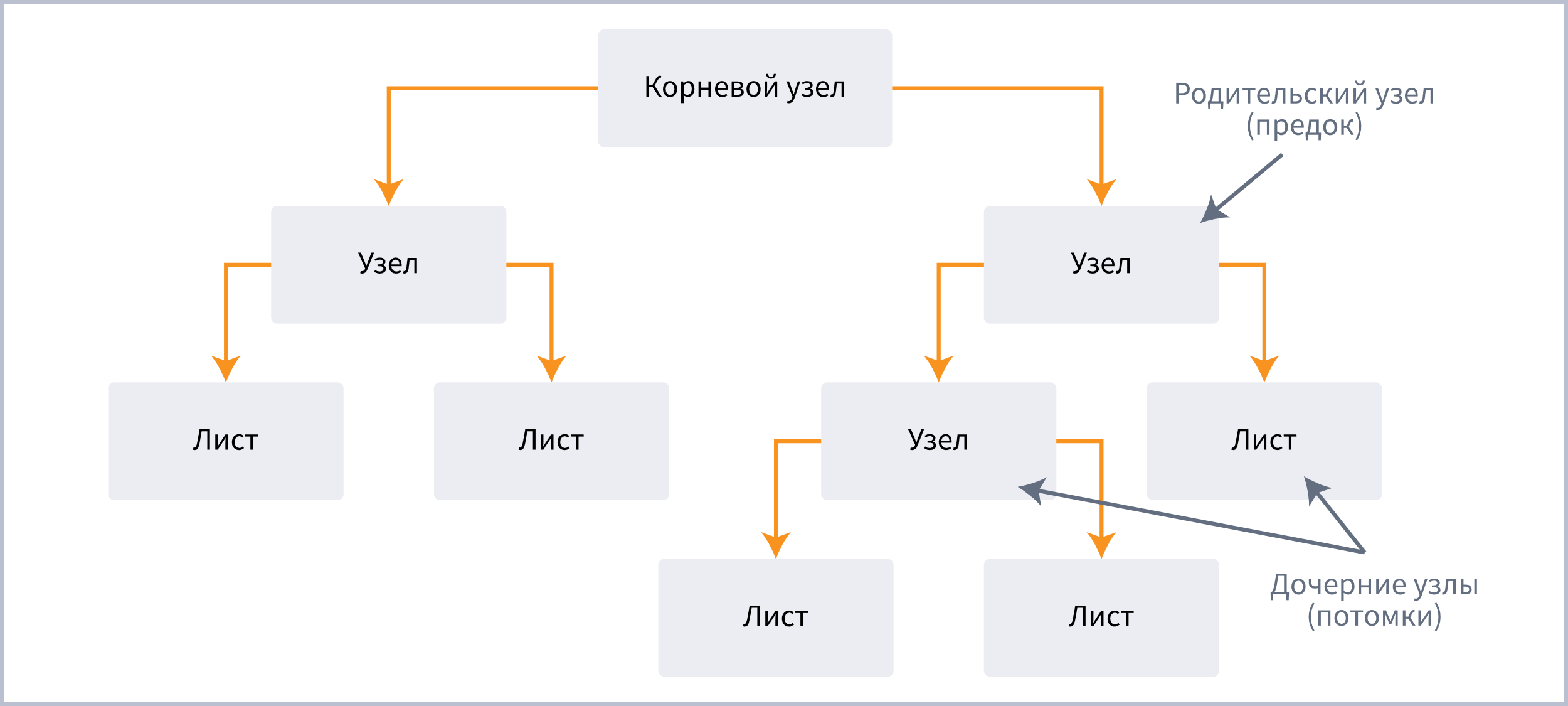


Рис. 15. Структура дерева решений

Значение метрики F1 полученное при классификации с помощью дерева решений равно 0.918.

## Результаты классификации с применением Логистической регрессии

Логистическая регрессия — метод построения линейного классификатора, позволяющий оценивать апостериорные вероятности принадлежности объектов классам. В данной работе был использован вариант классификатора для мультиклассовой классификации, когда необходимо предсказать принадлежность к более чем двум классам. В результате использование логистической регрессии было получено значение метрики F1 равное 0.918.

## Анализ полученных результатов кластеризации

Для кластеризации полученных параданных использовался алгоритм K-средних c различными параметрами инициализации и числа запусков. Изменение параметров инициализации не оказали существенное изменение значений метрик и результатов кластеризации. При оценке результатов кластеризации были использованы следующие метрики:

* inertia;
* homogeneity;
* completness;
* v-measure;
* ARI;
* AMI;
* silhouette.

Применив обработанные параданные в качестве входных данных для алгоритма кластеризации, мы получили следующие результаты кластеризации для 4 классов.

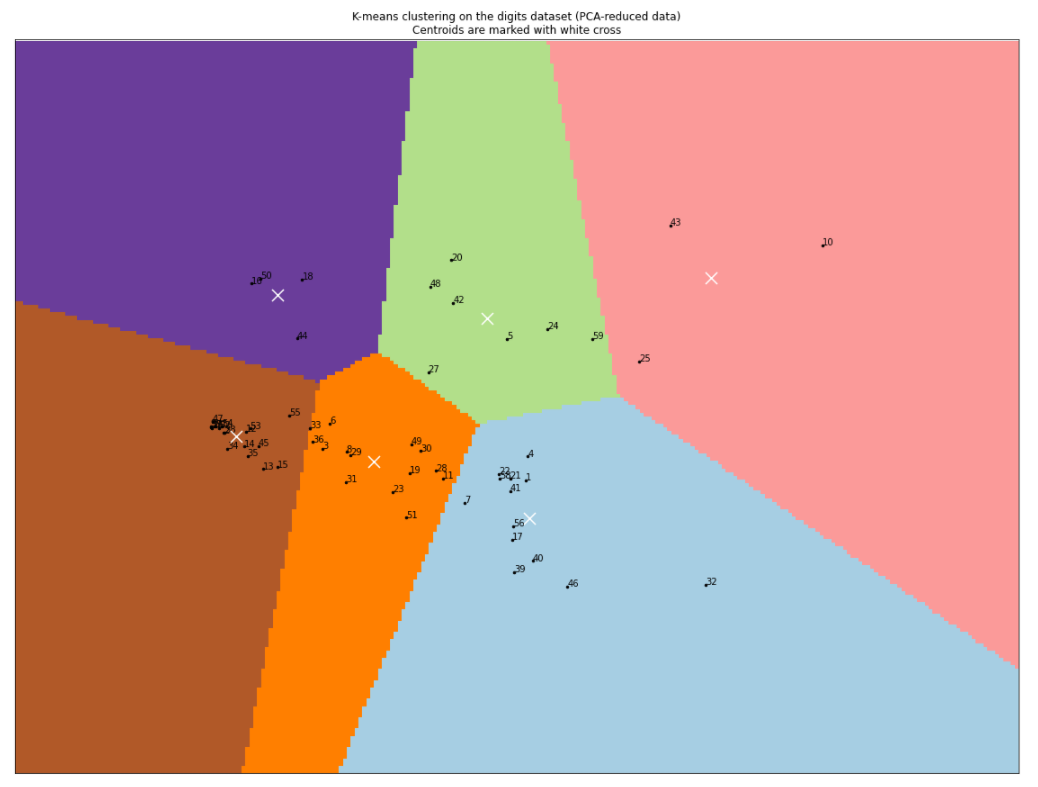


Рис. 16. Результаты кластеризации для n\_clusters = 4

Значения метрик при кластеризации и значении параметра n\_clusters = 4 представлены на рисунке 17.

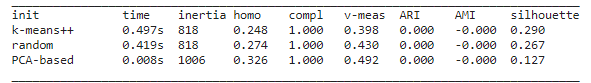


Рис.17. Результаты оценки эффективности кластеризации при n\_clusters = 4

Далее значение параметра n\_clusters было установлено равным 6. Результаты кластеризации при попытке выделить 6 различных классов приведены на рисунке ниже.

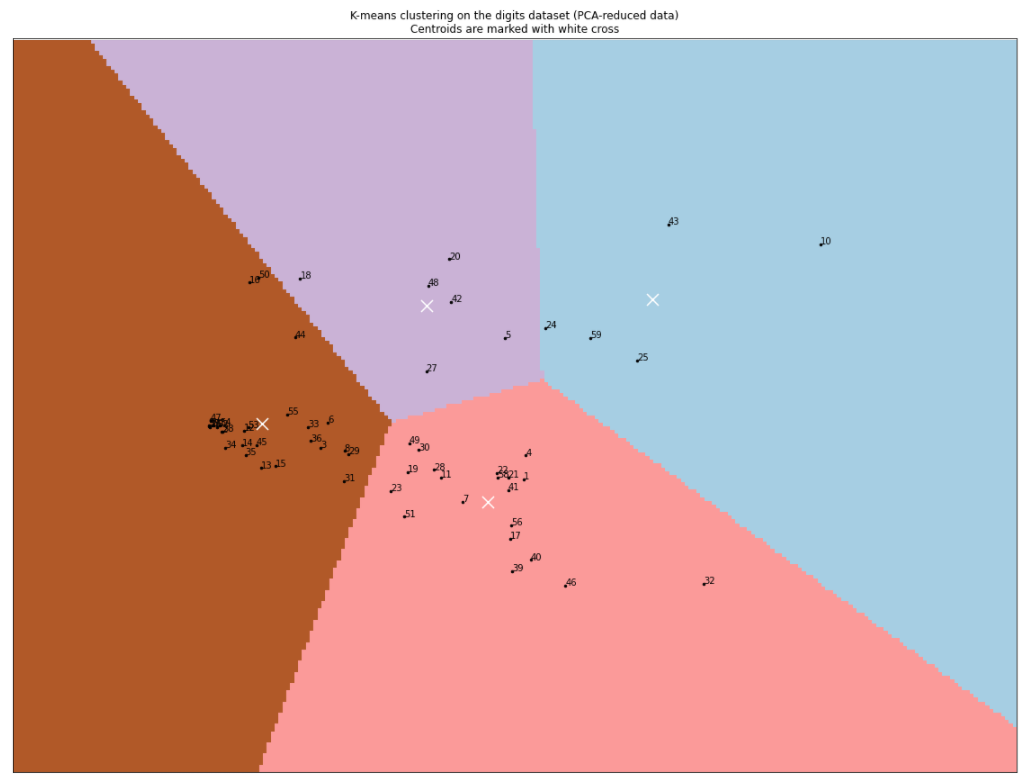
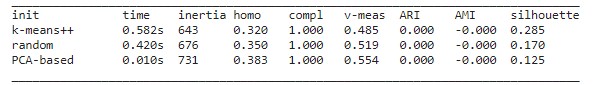


Рис. 18. Результаты кластеризации для n\_clusters = 6

Значения метрик при кластеризации и значении параметра n\_clusters = 6 представлены на рисунке 19.

Рис. 19. Результаты оценки эффективности кластеризации при n\_clusters = 6

Далее значение параметра n\_clusters было установлено равным 2. Результаты кластеризации при попытке выделить 2 различных класса приведены на рисунке ниже.

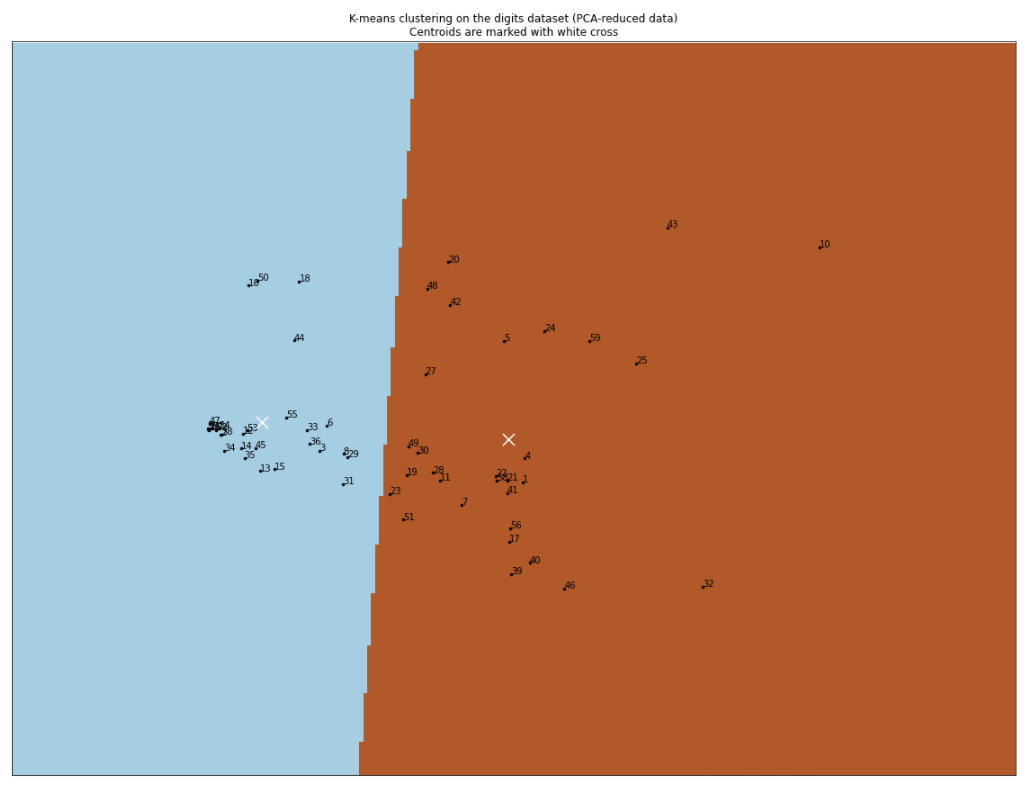


Рис. 20. Результаты кластеризации для n\_clusters = 2

Значения метрик при кластеризации и значении параметра n\_clusters = 2 представлены на рисунке 21.

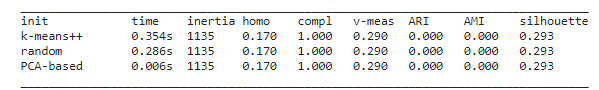


Рис. 21. Результаты оценки эффективности кластеризации при n\_clusters = 2

# 3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## 3.1. Общая информация о проведенном тестировании

Созданное для тестирования web-приложение было использовано для проведения тестирования, дальнейшего анализа и оценки собранной информации, как о знаниях респондентов по представленным в вопросах дисциплинах, так и о поведении респондентов в процессе ответов на представленные вопросы.

В качестве тестовых заданий были предложено 6 вопросов из различных разделов высшей математики, которые преподаются на 1 и 2 курсах бакалавриата. Число вопросов и правильных ответов на каждый вопрос представлено в таблице 2.

Таблица 2. Общее число ответов и число правильных ответов на вопросы теста.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Номер вопроса | 1 | 2 | 3 | 4 | 5 | 6 |
| Число вариантов ответа | 11 | 12 | 28 | 17 | 9 | 17 |
| Число правильных ответов | 6 | 5 | 2 | 4 | 6 | 5 |

Респондентам был предоставлен url с помощью которого они получали доступ к прохождению теста. Всего в тестировании приняло участие более 700 человек, но при этом стоит заметить, что на все вопросы теста ответили только 59 респондентов. Статистика по ответам респондентов на вопросы представлена в таблице 3.

Таблица 3. Число ответов респондентов на каждый из вопросов теста.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Номер вопроса |  | 1 | 2 | 3 | 4 | 5 | 6 |
| Число ответов респондентов |  | 143 | 91 | 73 | 72 | 71 | 59 |

Из приведенной таблицы видно, что после первого вопроса большое число респондентов прекратило проходить опрос. При обработке и анализе и данных использовались только данные респондентов, которые полностью прошли тест.

## Использование Google Colaboratory

Для работы с данными был выбран бесплатный облачный сервис Google Colaboratory (Colab), предоставляющий все необходимое для анализа данных и машинного обучения. Написание программного кода осуществляется в окне браузера в специальных блоках на странице проекта. Внешний вид страницы проекта представлен на рисунке 22.

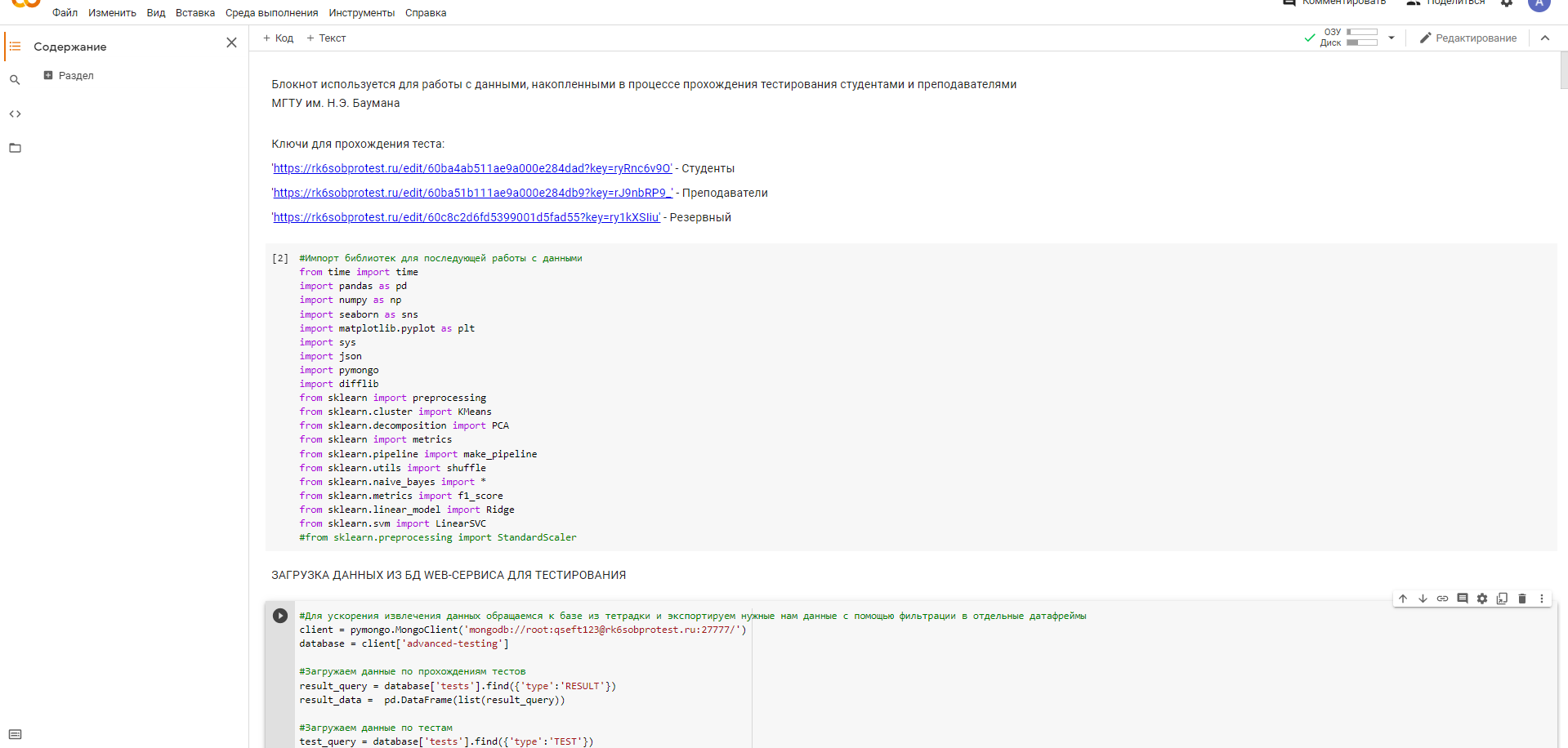


Рис.22. Внешний вид страницы проекта в сервисе Colab.

Страница с проектом для анализа собранных в процессе тестирования данных представляет собой набор различных блоков, содержащих в себе текст, программный код или их комбинации. Выбранный сервис поддерживает язык программирования Python версий 2 и 3, а также дает бесплатный доступ к быстрым GPU и TPU.

## Библиотека Pandas

Для работы с полученными данными в сервисе Colab была выбрана библиотека Pandas - высокоуровневая библиотека, построенная она поверх более низкоуровневой библиотеки NumPy (написана на Си), что является большим плюсом в производительности. В экосистеме Python, Pandas является наиболее продвинутой и быстроразвивающейся библиотекой для обработки и анализа данных. Библиотека имеет две главных структуры данных – Series и DataFrame.

Структура Series представляет из себя объект, похожий на одномерный массив, но отличительной его чертой является наличие ассоциированных меток(индексов), вдоль каждого элемента из списка. Такая особенность превращает его в ассоциативный массив или словарь в Python.

Структура DataFrame является табличной структурой данных. Данная структура имеет ассоциированные метки для строк и столбцов. Столбцами в объекте DataFrame выступают объекты Series, строки которых являются их непосредственными элементами. Вид структуры DataFrame представлен на рисунке.

## Seaborn для визуализации данных

Визуализация данных — это метод, который позволяет преобразовывать сырые данные в диаграммы и графики, которые несут ценную информацию. Диаграммы уменьшают сложность данных и делают более понятными для любого пользователя.

В качестве инструмента для визуализации тепловых карт взаимодействия пользователя с веб-страницей и для визуализации результатов кластеризации был выбран Seaborn – высокоуровневое API на базе библиотеки Matplotlib. Seaborn не уступает по функционалу библиотеке Matplotlib, но в то же время предоставляет большую абстракцию для упрощения графиков и привносит некоторые уникальные функции. Также в библиотеке есть достаточно сложные типы визуализации, которые в Matplotlib потребовали бы большого количество кода.

## Scikit-learn для машинного обучения

Для машинного обучения на Python существует много библиотек. В качестве библиотеки для работы с моделями машинного обучения была выбрана Scikit-Learn. Данная библиотека упрощает процесс создания классификатора и помогает более чётко выделить концепции машинного обучения, реализуя их с помощью понятной, хорошо документированной и надёжной библиотеки. Scikit-Learn содержит реализацию целого ряда алгоритмов для обучения с учителем (Supervised Learning) и обучения без учителя (Unsupervised Learning), а также предоставляет метрики для оценки использованных алгоритмов.

# ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной магистерской работы были исследованы методы анализа параданных при прохождении онлайн-тестов. Полученные результаты анализа параданных дают возможность сравнить их с общепринятой практикой оценки тестирования.

В ходе выполнения работы было разработано приложение, которое реализует необходимый функционал прохождения тестов и сбора параданных, также был проведен анализ полученных данных и реализован функционал для кластеризации и классификации результатов прохождения онлайн-теста. Система была развёрнута на облачном сервере, и было проведено её тестирование на студентах факультета РК МГТУ им. Н.Э. Баумана.

Проведенная работа и оценка классификации различными алгоритмами результатов прохождения теста студентами позволила выявить эффективность используемых алгоритмов классификации и правильность выбора признаков для формирования входных данных, не смотря на небольшой объем параданных.

Стоит отметить, что в рамках работы не было реализовано отображение результатов анализа параданных алгоритма в интерфейсе веб-приложения, так как анализ и просмотр результатов осуществлялся на стороннем облачном сервисе. Реализация данного фунционала и увеличение размеров набора входных данных для алгоритмов машинного обучения являются основными задачами для дальнейшей разработки.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Белоножко П.П., Карпенко А.П., Храмов Д.А. Анализ образовательных данных: направления и перспективы применения. //Интернет-журнал «НАУКОВЕДЕНИЕ». 2017. Том 9. № 4. [Электронный ресурс]. Режим доступа: http://naukovedenie.ru/PDF/15TVN417.pdf
2. Шмелева Е. Д., Семенова Т. В. Академическое мошенничество студентов учебная мотивация vs образовательная среда //Вопросы образования НИУ ВШЭ. Июль 2017.
3. Шмелева Е. Д. Академическое мошенничество в современных университетах: обзор теоретических подходов и результатов эмпирических исследований //Экономическая социология. Т. 16. No 2. Март 2015.
4. Безсмертная Е.Р. Академическое мошенничество в университетах: можно ли ему противодействовать? //ДИСКУССИЯ. Журнал научных публикаций. No 11(74). Декабрь 2016.
5. Леонтьев К.А., Панин С. Д., Холодный Ю. И. Оценка результатов тестирования на полиграфе методами регрессионного анализа. //Наука и Образование. МГТУ им. Н.Э. Баумана. Электрон. журн. 2014. No 10. С. 230–243. [Электронный ресурс] Режим доступа: http://engineering-science.ru/doc/728906.html
6. Щербина Д. Н. Повышение эффективности контроля знаний студентов на основании анализа последовательности решения тестовых заданий. //Образовательные технологии и общество. 2016.
7. Буре Н. А., Гребенникова Н.Л., Староверова К. Ю. Применение статистических методов при оценке качества учебного процесса. //Вестник Санкт-Петербургского университета. Т.14. Вып.4. 2018
8. Д.Н. Щербина. Стратегии прохождения тестов знаний, выявленные методом хронометрии просмотра вариантов ответа. //Валеология. 2015. No 4. с. 112-121.
9. Горлушкина Н.Н. Задачи и методы интеллектуального анализа образовательных данных для поддержки принятия решений. //Образовательные технологии и общество. 2015.
10. Rodrigues, M.W., Zárate, L.E., Isotani, S., Educational Data Mining: A review of evaluation process in the e-learning, Telematics and Informatics 2018. [Электронный ресурс] Режим доступа: https://doi.org/10.1016/j.tele.2018.04.015
11. Alyahyan E., Düştegör D. Predicting academic success in higher education: literature review and best practices. //International Journal of Educational Technology in Higher Education. 2020, 17:3
12. S. Lakshmi Prabha. Application of Educational Data mining techniques in e-Learning - A Case Study. // (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (5)
13. [Иванова Е. Е.](https://vestnik.guu.ru/index.php/jour/search?authors=Е.%20AND%20Е.%20AND%20Иванова) Образование в удаленном доступе: взгляд преподавателя. // ГУУ. Вестник университета. 2020(8) с.188-193.
14. Назарова Л.И., Симан А.С., Лямина И.М., Колоскова Г.А. Организация прокторинга в дистанционном обучении студентов аграрного вуза. // АГРОИНЖЕНЕРИЯ. 2020, No 4(98)
15. Баранников К.А., Лесин С.М. Методология анализа больших данных в образовании. //Народное образование. 2020, No 2.
16. Shrestha S., Pokharel M. Educational data mining in moodle data //International Journal of Informatics and Communication Technology (IJ-ICT) Vol.10, No.1, April 2021, pp. 9~18
17. Alisa Bilal Zorić. Benefits of Educational Data Mining. // Journal of International Business Research and Marketing. Volume 6, Issue 1, 2020
18. Python.org [Электронный ресурс] Режим доступа: https://python.org
19. MongoDB [Электронный ресурс] Режим доступа: https://mongodb.com
20. Pandas documentation [Электронный ресурс] Режим доступа: https://pandas.pydata.org
21. Seaborn documentation [Электронный ресурс] Режим доступа: https://seaborn.pydata.org
22. Google Colaboratory [Электронный ресурс] Режим доступа: https://colab.research.google.com
23. scikit-learn [Электронный ресурс] Режим доступа: https://scikit-learn.org
24. MachineLearning.ru [Электронный ресурс] Режим доступа: http://www.machinelearning.ru
25. Typescript [Электронный ресурс] Режим доступа: https://www.typescriptlang.org
26. Лебедев Д.В. Параданные: определение, типы, сбор и возможное применение //Мониторинг общественного мнения № 2 (156) март — апрель 2020

# ПРИЛОЖЕНИЕ А

# -\*- coding: utf-8 -\*-

"""My Master's\_thesis.ipynb

Automatically generated by Colaboratory.

Original file is located at <https://colab.research.google.com/drive/1OxMyxqgbmtmnx7bEWGbQjdiM0LXwwp-t>

Блокнот используется для работы с данными, накопленными в процессе прохождения тестирования студентами и преподавателями МГТУ им. Н.Э. Баумана

Ключи для прохождения теста:

'https://rk6sobprotest.ru/edit/60ba4ab511ae9a000e284dad?key=ryRnc6v9O' — Студенты

'https://rk6sobprotest.ru/edit/60ba51b111ae9a000e284db9?key=rJ9nbRP9\_' - Преподаватели

'https://rk6sobprotest.ru/edit/60c8c2d6fd5399001d5fad55?key=ry1kXSIiu' — Резервный

"""

#Импорт библиотек для последующей работы с данными

from time import time

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import sys

import json

import pymongo

import difflib

from sklearn import preprocessing

from sklearn.cluster import KMeans

from sklearn.decomposition import PCA

from sklearn import metrics

from sklearn.pipeline import make\_pipeline

from sklearn.utils import shuffle

from sklearn.naive\_bayes import \*

from sklearn.metrics import f1\_score

from sklearn.metrics import mean\_squared\_error

from sklearn.linear\_model import Ridge

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import LinearSVC

from math import sqrt

#from math import max#

#from sklearn.preprocessing import StandardScaler

"""ЗАГРУЗКА ДАННЫХ ИЗ БД WEB-СЕРВИСА ДЛЯ ТЕСТИРОВАНИЯ"""

#Для ускорения извлечения данных обращаемся к базе из тетрадки и экспортируем нужные нам данные с помощью фильтрации в отдельные датафреймы

client = pymongo.MongoClient('mongodb://root:qseft123@rk6sobprotest.ru:27777/')

database = client['advanced-testing']

#Загружаем данные по прохождениям тестов

result\_query = database['tests'].find({'type':'RESULT'})

result\_data = pd.DataFrame(list(result\_query))

#Загружаем данные по тестам

test\_query = database['tests'].find({'type':'TEST'})

test\_data = pd.DataFrame(list(test\_query))

#Загружаем данные по заданиям

task\_query = database['tests'].find({'type':'TASK'})

task\_data = pd.DataFrame(list(task\_query))

#Загружаем данные по ответам

answer\_query = database['tests'].find({'type':'ANSWER'})

answer\_data = pd.DataFrame(list(answer\_query))

#Загружаем данные по событиям во время ответа на вопросы

task\_event\_query = database['tests'].find({'type':'TASK\_EVENT'})

task\_event\_data = pd.DataFrame(list(task\_event\_query))

#Загружаем данные по глобальным событиям в процессе всего тестирования

global\_event\_query = database['tests'].find({'type':'GLOBAL\_EVENT'})

global\_event\_data = pd.DataFrame(list(global\_event\_query))

global\_event\_data.head()

task\_event\_data['eventType'].unique()

task\_event\_data.loc[(task\_event\_data['eventType'] == 'MOUSE\_CLICK') & (task\_event\_data['resultId'] == '60c87523fd5399001d5faa03')]

global\_event\_data['eventType'].unique()

# Commented out IPython magic to ensure Python compatibility.

# %%capture

# #Фукнция для изменения формата данных на нужный нам

# def convertFromBArray(row):

# row.\_id = str(row.\_id)

# return row

#Приводим все id к строчному виду

result\_data = result\_data.apply(convertFromBArray, axis='columns')

test\_data = test\_data.apply(convertFromBArray, axis='columns')

task\_data = task\_data.apply(convertFromBArray, axis='columns')

answer\_data = answer\_data.apply(convertFromBArray, axis='columns')

task\_event\_data = task\_event\_data.apply(convertFromBArray, axis='columns')

global\_event\_data = global\_event\_data.apply(convertFromBArray, axis='columns')

#Извлекаем только записи результата, в которых пользователь закончил тест и ответил на все вопросы

result\_data = result\_data.loc[result\_data['isFinished'] != False]

for index, row in result\_data.iterrows():

lenarr = row['answerIds']

if len(lenarr) < 6:

result\_data.drop(index, inplace=True)

result\_data.index = np.arange(1, len(result\_data) + 1)

result\_data.info()

answer\_data.info()

#answer\_data.iloc[answer\_data.groupby('taskId').taskId.transform('size').argsort(kind='mergesort')]

answer\_data.groupby('taskId').agg(['count'])

#df[['col1', 'col2', 'col3', 'col4']].groupby(['col1', 'col2']).agg(['mean', 'count'])

kek = test\_data['taskIds'][0]

task\_data.loc[task\_data['\_id'].isin(kek)]

"""ПОЛУЧАЕМ id ТЕСТОВ"""

def getTestNameId(df):

'''Сохраняем в словарь id и названия тестов'''

testNameId = dict()

for index, row in df.iterrows():

testNameId[row['name']] = str(row['\_id'])

return testNameId

#Получаем названия тестов и их id

TestNameAndId = getTestNameId(test\_data)

#TestNameAndId

"""ПОЛУЧАЕМ id ПРОХОЖДЕНИЙ ТЕСТОВ"""

def getTestResults(df, testId = False):

'''Сохраняем в массив resultIds id проходов теста, а в массив answersIds id ответов при соответствующем проходе теста'''

resultIds = []

answerIds = []

for index, row in df.iterrows():

if testId:

if row['testId'] == testId:

resultIds.append(str(row['\_id']))

answerIds.append(row['answerIds'])

else:

resultIds.append(str(row['\_id']))

answerIds.append(row['answerIds'])

return resultIds, answerIds

#Получаем id проходов теста и id ответов

resultIds, answerIds = getTestResults(result\_data, TestNameAndId['Тест для студентов МГТУ им Н.Э. Баумана'])

#resultIds

#resultIds, answerIds = getTestResults(result\_data, TestNameAndId['Тест для преподавателей МГТУ им Н.Э. Баумана'])

#resultIds

"""ПОЛУЧАЕМ СПИСОК ТЕСТОВЫХ ВОПРОСОВ И ПРАВИЛЬНЫХ ОТВЕТОВ К НИМ"""

def getAnswersForTests(test\_df, task\_df, testId):

'''Сохраняем в словарь вопросы и правильные ответы в виде датафрейма '''

answersAndQuestions = dict()

taskIds = []

test\_df = test\_df.loc[test\_df['\_id'] == testId]

qa\_df = task\_df.loc[task\_df['\_id'].isin(test\_df.taskIds[0])]

qa\_df = qa\_df[['\_id', 'question', 'correctAnswers']]

return qa\_df

#Получаем датафрейм с вопросами и правильными ответами на них

questionsAndAnswersStd = getAnswersForTests(test\_data, task\_data, TestNameAndId['Тест для студентов МГТУ им Н.Э. Баумана'])

#questionsAndAnswersStd

#Создаем словарь для хранения значений будущего датафрейма для кластеризации

def createDictKeys(answsize):

keysArr = []

for i in range(1, answsize + 1):

keysArr.append(str(i) + '\_lt')

keysArr.append(str(i) + '\_mc')

keysArr.append(str(i) + '\_сb')

keysArr.append(str(i) + '\_cbu')

keysArr.append('resize')

keysArr.append('console')

keysArr.append('focus')

return keysArr

def evaluateAnswer(correctArr, userArr):

pass

corrArrLen = len(correctArr)

correctArr = correctArr.sort()

userArr = userArr.sort()

countSameAnswers = np.sum(correctArr == userArr)

return float(countSameAnswers) / float(corrArrLen)

questionsAndAnswersStd.shape

"""НАЧИНАЕМ ФОРМИРОВАТЬ ДАТАФРЕЙМ С ДАННЫМИ ПО КАЖДОМУ ПРОХОЖДЕНИЮ ТЕСТА"""

def createFeatureDF(result\_id, answers\_id, answer\_df, task\_df, qanda\_dict, task\_event\_df):

keysArr = createDictKeys(qanda\_dict.shape[0])

dictForDF = dict([(key, []) for key in keysArr])

for i in range(len(answers\_id)):

tmp\_df = answer\_df.loc[answer\_df['\_id'].isin(answers\_id[i])]

tmp\_df = pd.merge(tmp\_df, task\_df, left\_on= 'taskId', right\_on='\_id', suffixes=('a', 't'))

changeAnswer\_count = tmp\_df.shape[0]

res\_df = tmp\_df.drop\_duplicates(subset=['\_idt'], keep='last')

res\_df.index = np.arange(1, len(res\_df) + 1)

if res\_df.shape[0] < 6:

continue

#display(res\_df)

for index, row in res\_df.iterrows():

ltIndex = str(index) + '\_lt'

mcIndex = str(index) + '\_mc'

cbIndex = str(index) + '\_сb'

cbuIndex = str(index) + '\_cbu'

tmpEventsArr = row['eventIds']

tmpEventsDf = task\_event\_df.loc[task\_event\_df['\_id'].isin(tmpEventsArr)]

countClick = tmpEventsDf.loc[tmpEventsDf['eventType'] == 'MOUSE\_CLICK'].shape[0]

countCb = tmpEventsDf.loc[tmpEventsDf['eventType'] == 'CHECKBOX\_CHECKED'].shape[0]

#print(countCb)

countCbu = tmpEventsDf.loc[tmpEventsDf['eventType'] == 'CHECKBOX\_UNCHECKED'].shape[0]

dictForDF[ltIndex].append(row['time'])

dictForDF[mcIndex].append(countClick)

dictForDF[cbIndex].append(countCb)

dictForDF[cbuIndex].append(countCbu)

return dictForDF

baseFeatureDict = createFeatureDF(resultIds, answerIds, answer\_data, task\_data, questionsAndAnswersStd, task\_event\_data)

def processTaskEvents(featureDict, resultIds, taskEventsData, globalEventsData):

#print(resultIds)

#print(featureDict)

for res in resultIds:

taskData = taskEventsData.loc[taskEventsData['resultId'] == res]

globalData = globalEventsData.loc[globalEventsData['resultId'] == res]

#featureDict['click'].append(taskData.loc[taskData['eventType'] == 'MOUSE\_CLICK'].shape[0])

#featureDict['checkbox'].append(taskData.loc[taskData['eventType'] == 'CHECKBOX\_CHECKED'].shape[0])

#featureDict['checkbox\_u'].append(taskData.loc[taskData['eventType'] == 'CHECKBOX\_UNCHECKED'].shape[0])

featureDict['resize'].append(globalData.loc[globalData['eventType'] == 'RESIZE'].shape[0])

featureDict['console'].append(globalData.loc[globalData['eventType'] == 'CONSOLE\_OPENED'].shape[0])

featureDict['focus'].append(globalData.loc[globalData['eventType'] == 'FOCUS\_OUT'].shape[0])

return featureDict

featureDictWithEventsData = processTaskEvents(baseFeatureDict, resultIds, task\_event\_data, global\_event\_data)

#featureDictWithEventsData

model = pd.DataFrame(data=featureDictWithEventsData)

model

model.describe()

#делаем копию датафрейма для использования в классификаторе

#Пробуем разметить данные для обучения.

# 1 - нет переключений, длительный просмотри вопросов

# 2 - есть переключения, длительный просмотр ворпросов

# 3 - нет переключений, короткий просмотр вопросов

# 4 - есть переключения, короткий просмотр

# 5

# 6

classificationModel = shuffle(model, random\_state = 1)

classificationModel = classificationModel.reset\_index(drop = True)

X\_train = classificationModel[:50]

X\_test = classificationModel[50:]

X\_test = X\_test.values

X\_train

y\_trainDict =[3, 4, 4, 2, 2, 3, 2, 2, 4, 1, 3, 3, 2, 1, 4, 2, 2, 3, 2, 4, 2, 3, 4, 3, 2, 2, 4, 4, 4, 1, 3, 4]

y\_train = pd.DataFrame(data = y\_trainDict)

classificationModel

X\_train

#y\_test = [4, 4, 4, 3, 3, 3, 3, 2, 3, 2, 2, 3, 2, 2, 2]

#X\_train = X\_train.values

np.array(classificationModel['1\_lt'])

#Пишем функцию для автоматической простановки классов нашим обучающим данным

# Всего мы будем иметь 6 классов:

# 2 варианта по наличию или отсутствию признаков списывания

# 3 варианта по времени просмотра тестового задания

def datasetLabeling(model):

labels = []

mean = []

res\_mse = []

for i in range(1, 7):

index = str(i) + '\_lt'

mean.append(model[index].mean())

minusmean = model[index] - mean[i - 1]

mse = 0

for value in minusmean:

value = value \*\* 2

mse += value

mse = sqrt(mse / len(minusmean))

res\_mse.append(mse)

#print(mean)

#print(res\_mse)

left\_b = []

right\_b = []

for j in range(len( mean)):

left\_b.append(abs(mean[j] - (0.01 \* res\_mse[j])))

right\_b.append(abs(mean[j] + (0.01 \* res\_mse[j])))

#return model, labels

#print(left\_b)

#print(right\_b)

cheat\_c = np.array(model['console'] + model['resize'] + model['focus'])

#print(kek)

for i, row in model.iterrows():

count = [0, 0, 0]

for k in range(1, 7):

index = str(k) + '\_lt'

if left\_b[k - 1] <= row[index] <= right\_b[k - 1]: #внутри границы mean +- sigma

count[0] += 1

elif left\_b[k - 1] > row[index]: #слева от границы

count[1] += 1

else: #справа от границы

count[2] += 1

rescount = count.index(max(count))

#print(rescount)

if cheat\_c[i] > 5 and (rescount == 0):

labels.append(1)

elif cheat\_c[i] > 5 and (rescount == 1):

labels.append(2)

elif cheat\_c[i] > 5 and (rescount == 2):

labels.append(3)

elif cheat\_c[i] <= 5 and (rescount == 0):

labels.append(4)

elif cheat\_c[i] <= 5 and (rescount == 1):

labels.append(5)

else:

labels.append(6)

return labels

labels = datasetLabeling(classificationModel)

X\_train = classificationModel[:30]

X\_test = classificationModel[30:]

X\_test = X\_test.values

#X\_train

y\_train = labels[0:30]

y\_test = labels[30:]

X\_train = X\_train.values #returns a numpy array

min\_max\_scaler = preprocessing.MinMaxScaler()

X\_train = min\_max\_scaler.fit\_transform(X\_train)

#X\_train

#normalizedModel = pd.DataFrame(data\_scaled)

#normalizedModel = model

y\_test

#Пробуем использовать наивный баесовский классификатор

#

clf = BernoulliNB()

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

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

y\_pred

print(f1\_score(y\_test, y\_pred, average='micro'))

ridge\_clf = Ridge()

ridge\_clf.fit(X\_train, y\_train)

ridge\_y\_pred = clf.predict(X\_test)

ridge\_y\_pred

print(f1\_score(y\_test, ridge\_y\_pred, average='micro'))

svm\_clf = LinearSVC(dual= False)

svm\_clf.fit(X\_train, y\_train)

svm\_y\_pred = clf.predict(X\_test)

svm\_y\_pred

print(f1\_score(y\_test, svm\_y\_pred, average='micro'))

dt\_clf = DecisionTreeClassifier( random\_state=0)

dt\_clf.fit(X\_train, y\_train)

dt\_y\_pred = clf.predict(X\_test)

dt\_y\_pred

lr\_clf = LogisticRegression( random\_state=0)

lr\_clf.fit(X\_train, y\_train)

lr\_y\_pred = clf.predict(X\_test)

lr\_y\_pred

#Подготавливаем данные для модели

#Нормализация

#data = model.values #returns a numpy array

data = classificationModel.values

min\_max\_scaler = preprocessing.MinMaxScaler()

data\_scaled = min\_max\_scaler.fit\_transform(data)

normalizedModel = pd.DataFrame(data\_scaled)

#normalizedModel = model

labels = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]

n\_clusters = 4

def bench\_k\_means(kmeans, name, data, labels):

"""Benchmark to evaluate the KMeans initialization methods.

Parameters

----------

kmeans : KMeans instance

A :class:`~sklearn.cluster.KMeans` instance with the initialization

already set.

name : str

Name given to the strategy. It will be used to show the results in a

table.

data : ndarray of shape (n\_samples, n\_features)

The data to cluster.

labels : ndarray of shape (n\_samples,)

The labels used to compute the clustering metrics which requires some

supervision.

"""

t0 = time()

estimator = make\_pipeline(preprocessing.StandardScaler(), kmeans).fit(data)

fit\_time = time() - t0

results = [name, fit\_time, estimator[-1].inertia\_]

# Define the metrics which require only the true labels and estimator

# labels

clustering\_metrics = [

metrics.homogeneity\_score,

metrics.completeness\_score,

metrics.v\_measure\_score,

metrics.adjusted\_rand\_score,

metrics.adjusted\_mutual\_info\_score,

]

results += [m(labels, estimator[-1].labels\_) for m in clustering\_metrics]

# The silhouette score requires the full dataset

results += [

metrics.silhouette\_score(data, estimator[-1].labels\_,

metric="euclidean", sample\_size=300,)

]

# Show the results

formatter\_result = ("{:9s}\t{:.3f}s\t{:.0f}\t{:.3f}\t{:.3f}"

"\t{:.3f}\t{:.3f}\t{:.3f}\t{:.3f}")

print(formatter\_result.format(\*results))

print(82 \* '\_')

print('init\t\ttime\tinertia\thomo\tcompl\tv-meas\tARI\tAMI\tsilhouette')

kmeans = KMeans(init="k-means++", n\_clusters= n\_clusters, n\_init=4,

random\_state=0)

bench\_k\_means(kmeans=kmeans, name="k-means++", data=normalizedModel, labels=labels)

kmeans = KMeans(init="random", n\_clusters= n\_clusters, n\_init=4, random\_state=0)

bench\_k\_means(kmeans=kmeans, name="random", data=normalizedModel, labels=labels)

pca = PCA(n\_components=n\_clusters).fit(normalizedModel)

kmeans = KMeans(init=pca.components\_, n\_clusters= n\_clusters, n\_init=1)

bench\_k\_means(kmeans=kmeans, name="PCA-based", data=normalizedModel, labels=labels)

print(82 \* '\_')

reduced\_data = PCA(n\_components=2).fit\_transform(normalizedModel)

kmeans = KMeans(init="random", n\_clusters=n\_clusters, n\_init=4)

kmeans.fit(reduced\_data)

# Step size of the mesh. Decrease to increase the quality of the VQ.

h = .02 # point in the mesh [x\_min, x\_max]x[y\_min, y\_max].

# Plot the decision boundary. For that, we will assign a color to each

x\_min, x\_max = reduced\_data[:, 0].min() - 1, reduced\_data[:, 0].max() + 1

y\_min, y\_max = reduced\_data[:, 1].min() - 1, reduced\_data[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))

# Obtain labels for each point in mesh. Use last trained model.

Z = kmeans.predict(np.c\_[xx.ravel(), yy.ravel()])

# Put the result into a color plot

Z = Z.reshape(xx.shape)

#plt.figure(1)

plt.figure(figsize=(20, 15))

plt.clf()

plt.imshow(Z, interpolation="nearest",

extent=(xx.min(), xx.max(), yy.min(), yy.max()),

cmap=plt.cm.Paired, aspect="auto", origin="lower")

plt.plot(reduced\_data[:, 0], reduced\_data[:, 1], 'k.', markersize=5)

# Plot the centroids as a white X

centroids = kmeans.cluster\_centers\_

plt.scatter(centroids[:, 0], centroids[:, 1], marker="x", s=169, linewidths=3,

color="w", zorder=10)

plt.title("K-means clustering on the digits dataset (PCA-reduced data)\n"

"Centroids are marked with white cross")

for i, txt in enumerate(labels):

plt.annotate(txt, (reduced\_data[i, 0], reduced\_data[i, 1]))

plt.xlim(x\_min, x\_max)

plt.ylim(y\_min, y\_max)

plt.xticks(())

plt.yticks(())

#plt.scatter()

plt.show()

reduced\_data

#y = [2.56422, 3.77284, 3.52623, 3.51468, 3.02199]

#z = [0.15, 0.3, 0.45, 0.6, 0.75]

#n = [58, 651, 393, 203, 123]

#fig, ax = plt.subplots()

#ax.scatter(z, y)

#for i, txt in enumerate(n):

# ax.annotate(txt, (z[i], y[i]))

from sklearn.datasets import load\_digits

data, labels = load\_digits(return\_X\_y=True)

result\_data['answerIds'][1]

kek = answer\_data.loc[answer\_data['\_id'].isin(answerIds[3])]

#new = kek['eventIds'][0]

#task\_event\_data.loc[task\_event\_data['\_id'].isin(new)]

#task\_event\_data.loc[task\_event\_data['\_id'].isin(kek['eventIds'][0])]

located\_mouse\_events\_ids = task\_event\_data.loc[(task\_event\_data['eventType'] == 'MOUSE\_CLICK') | (task\_event\_data['eventType'] == 'MOUSE\_MOVE')]

#located\_mouse\_events\_ids = task\_event\_data.loc[(task\_event\_data['eventType'] == 'MOUSE\_CLICK')]

#located\_mouse\_events\_ids = task\_event\_data.loc[(task\_event\_data['eventType'] == 'MOUSE\_MOVE')]

#located\_mouse\_events\_ids = located\_mouse\_events\_ids[located\_mouse\_events\_ids['\_id'].isin(new)]

#located\_mouse\_events\_ids['timestamp'][0]

#located\_mouse\_events\_ids['timestamp'] = pd.to\_datetime(located\_mouse\_events\_ids['timestamp'], format='%Y%m%d.0')

#for index, row in located\_mouse\_events\_ids.iterrows():

# if row['\_id'] not in new:

# located\_mouse\_events\_ids.drop(row)

#new

located\_mouse\_events\_ids

#Получаем набор координат во время ответа пользователя на первый вопрос.

resDfId = result\_data['\_id'][40]

resDfId

answerDfId = result\_data['answerIds'][1][0]

#answerDfId

answerDf = answer\_data.loc[answer\_data['\_id'] == answerDfId]

#answerDf

answerDfevents = answerDf['eventIds']

#answerDfevents

plotData = task\_event\_data.loc[task\_event\_data['resultId'] == '60cdde43cf30eb000e1dfdbe']

located\_mouse\_events\_ids = plotData

plotData

#located\_mouse\_events\_ids['value'].apply(pd.Series)

#located\_mouse\_events\_ids

#%%capture

located\_mouse\_events\_ids = pd.concat([located\_mouse\_events\_ids.drop(['value'], axis=1), located\_mouse\_events\_ids['value'].apply(pd.Series)], axis=1)

#located\_mouse\_events\_ids = located\_mouse\_events\_ids.loc[located\_mouse\_events\_ids['resultId'] == '60c8e4c6fd5399001d5fada5']

fig, ax = plt.subplots()

plt.axis([min(located\_mouse\_events\_ids['x']), max(located\_mouse\_events\_ids['x']), max(located\_mouse\_events\_ids['y']), min(located\_mouse\_events\_ids['y'])])

sns.kdeplot(located\_mouse\_events\_ids['x'],located\_mouse\_events\_ids['y'], shade=True, n\_levels=10, color='red')

plt.show()

fig, ax = plt.subplots()

plt.axis([min(located\_mouse\_events\_ids['x']), max(located\_mouse\_events\_ids['x']), max(located\_mouse\_events\_ids['y']), min(located\_mouse\_events\_ids['y'])])

sns.pointplot(located\_mouse\_events\_ids['x'],located\_mouse\_events\_ids['y'])

plt.show()

fig, ax = plt.subplots()

plt.axis([min(located\_mouse\_events\_ids['x']), max(located\_mouse\_events\_ids['x']), max(located\_mouse\_events\_ids['y']), min(located\_mouse\_events\_ids['y'])])

sns.scatterplot(located\_mouse\_events\_ids['x'],located\_mouse\_events\_ids['y'])

plt.show()

plt.hist2d(located\_mouse\_events\_ids['x'], located\_mouse\_events\_ids['y'], bins=100)

plt.show()

import numpy as np

import numpy.random

import matplotlib.pyplot as plt

import sphviewer as sph

def myplot(x, y, nb=32, xsize=500, ysize=500):

xmin = np.min(x)

xmax = np.max(x)

ymin = np.min(y)

ymax = np.max(y)

x0 = (xmin+xmax)/2.

y0 = (ymin+ymax)/2.

pos = np.zeros([len(x),3])

pos[:,0] = x

pos[:,1] = y

w = np.ones(len(x))

P = sph.Particles(pos, w, nb=nb)

S = sph.Scene(P)

S.update\_camera(r='infinity', x=x0, y=y0, z=0,

xsize=xsize, ysize=ysize)

R = sph.Render(S)

R.set\_logscale()

img = R.get\_image()

extent = R.get\_extent()

for i, j in zip(xrange(4), [x0,x0,y0,y0]):

extent[i] += j

print(extent)

return img, extent

fig = plt.figure(1, figsize=(10,10))

ax1 = fig.add\_subplot(221)

ax2 = fig.add\_subplot(222)

ax3 = fig.add\_subplot(223)

ax4 = fig.add\_subplot(224)

# Generate some test data

x = np.random.randn(1000)

y = np.random.randn(1000)

#Plotting a regular scatter plot

ax1.plot(x,y,'k.', markersize=5)

ax1.set\_xlim(-3,3)

ax1.set\_ylim(-3,3)

heatmap\_16, extent\_16 = myplot(x,y, nb=16)

heatmap\_32, extent\_32 = myplot(x,y, nb=32)

heatmap\_64, extent\_64 = myplot(x,y, nb=64)

ax2.imshow(heatmap\_16, extent=extent\_16, origin='lower', aspect='auto')

ax2.set\_title("Smoothing over 16 neighbors")

ax3.imshow(heatmap\_32, extent=extent\_32, origin='lower', aspect='auto')

ax3.set\_title("Smoothing over 32 neighbors")

#Make the heatmap using a smoothing over 64 neighbors

ax4.imshow(heatmap\_64, extent=extent\_64, origin='lower', aspect='auto')

ax4.set\_title("Smoothing over 64 neighbors")

plt.show()