МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ»

Факультет физико-математических и естественных наук

Кафедра информационных технологий

«Допустить к защите»

Заведующий кафедрой
информационных технологий
д.ф.-м.н.
_____ Ю.Н. Орлов
«____»_____ 20__Г.

Выпускная квалификационная работа бакалавра

Направление 09.03.03 «Прикладная информатика»

ТЕМА «Разработка информационной системы для анализа активности пользователей интернет портала онлайн-образования»

Выполнил студент **Еременко Артем Геннадьевич**

(Фамилия, имя, отчество)

Группа НПИбд-01-18

Студ. билет № 1032162043

Руководитель выпускной квалификационной работы

Хачумов М. В., к.ф.-м.н., доцент кафедры информационных технологий
(Ф.И.О., степень, звание, должность)

_____
(Подпись)

Автор _____
(Подпись)

г. Москва

2022 г.

Федеральное государственное автономное образовательное учреждение высшего образования
«Российский университет дружбы народов»

АННОТАЦИЯ

выпускной квалификационной работы

Еременко Артем Геннадьевич
(фамилия, имя, отчество)

на тему: «Разработка информационной системы для анализа активности пользователей интернет портала онлайн-образования»

Настоящее исследование затрагивает вопросы разработки информационной системы онлайн-курсов, а также возможности применения в ней современных методов анализа активности пользователей, в частности для расчёта успеваемости студентов. В данной работе описывается техническое задание по разработке модели информационной системы на основе UML. Также проводится обзор выскоуровневого языка графического описания UML: изучается то, какие строительные блоки в нём присутствуют, и как их следует применять; анализируются общие механизмы действия. Далее выполняется практическая часть диплома в виде построения одиннадцати диаграмм и архитектуры информационной системы интернет-портала онлайн-образования: 4 диаграммы вариантов использования, 4 диаграммы классов и 3 диаграммы последовательности действий для отдельно взятых вариантов использования. Во второй части работы разбираются методы машинного обучения и способы их применения для решения практических задач. Проводятся исследования с применением разведочного анализа данных (exploratory data analysis) и методов машинного обучения.

Оглавление

Список сокращений

Русскоязычные сокращения

ПС – программные системы

ОО – объектно-ориентированный

ИС – информационная система

ТУИС – телекоммуникационная учебно-информационная система

Англоязычные сокращения

UML – Unified Modeling Language

OMG – Object Management Group

EDA – Exploratory Data Analysis

SVM – Support Vector Machines

k-NN – k-nearest neighbors

CSV – Comma-Separated Values

Введение

В настоящее время, с развитием IT сферы, многие рутинные и организационные процессы плавно модернизируются. Это касается и системы образования, где с каждым годом начинают применять более новые подходы к способам преподнесения знаний, а также их проверки. В высших учебных заведениях уже применяют телекоммуникационные учебно-информационные системы (ТУИС) для размещения материалов и сбора лабораторных работ. Однако в ближайшей перспективе возможен частичны или даже полный переход обучения в онлайн-формат. Для осуществления этого процесса придётся разрабатывать множество информационных систем (ИС), построение которых невозможно без системного подхода к проектированию. Для этих целей хорошо подходит UML (Unified Modeling Language).

UML – это унифицированный графический язык моделирования для описания, визуализации, проектирования и документирования ОО (объектно-ориентированных) систем [1]. ULM служит для поддержки процессов моделирования ПС (программных систем) на основе ОО подхода, организовывать взаимосвязь концептуальных и программных понятий, отражать проблемы масштабирования сложных систем. Модели на UML используются на всех этапах жизненного цикла ПС, начиная с бизнес-анализа и заканчивая сопровождением системы. Разные организации могут применять UML по своему усмотрению в зависимости от своих проблемных областей и используемых технологий.

Одним из дополнительных и информативных компонентов онлайн-систем является модуль по анализу активности пользователей с применением машинного обучения для последующего использования полученной информации при оптимизации различных процессов.

Актуальность темы

Актуальность задачи заключается в возросшей необходимости разработки средств онлайн-образования в связи со сложившейся в мире сложной эпидемиологической обстановкой, из-за которой достаточно внушительная часть

студентов утратила возможность проходить очное обучение непосредственно в учебном заведении.

Также увеличились интерес и спрос на прохождении курсов по освоению современных профессий от коммерческих организаций, предоставляющих услуги в сфере онлайн-образования, по типу Coursera, Skill Box, Geek Brains, Нетология и т.п.

Цель работы

Данная дипломная работа посвящена решению современной задачи по разработке ИС интернет-портала онлайн-образования и анализу активности её пользователей с целью выявления закономерностей их поведения.

Методы исследования

В данной дипломной работе для решения задачи анализа активностей пользователей интернет-портала применяются методы EDA и машинного обучения с учителем.

В качестве источников информации использовалась российская и зарубежная научная периодика, а также ресурсы сети интернет.

Структура работы

В первой главе дана постановка задачи разработки ИС и анализа активности пользователей ИС, выполнен обзор языка UML и разработаны основные диаграммы.

Во второй главе исследованы и формально описаны современные методы машинного обучения для решения задачи анализа активности пользователей информационной системы.

В третьей главе предложены методы и алгоритмы решения задачи поставленной задачи.

В четвертой главе исследуются практические вопросы решения задачи, проводятся экспериментальные исследования.

1 Постановка задачи и методы разработки информационных системы

ИС предназначена для прохождения онлайн-занятий и сдачи домашних заданий по выбранному направлению подготовки. Дополнительно требуется разработать специальную подсистему для анализа активности пользователей. У пользователя есть возможность проходить одновременно несколько курсов.

Ниже представлены техническое задание по разработке ИС, обзор и исследование языка моделирования UML, а также разработаны UML диаграммы и архитектура ИС.

1.1 Постановка задачи разработки информационной системы

В первую очередь, требуется составить техническое задание для того, чтобы иметь четкие критерии оценки результата работы. Далее необходимо сделать обзор UML для создания представления об описании, визуализации, проектировании и документировании ИС. В конечном счёте должна быть разработана модель информационной системы интернет-портала онлайн-образования на основе UML.

Техническое задание

Информационная система предназначена для прохождения онлайн-занятий по освоению выбранного курса обучения.

Пользователи системы:

1. Неавторизованные посетители сайта (Посетитель)
2. Авторизованные студенты (Студент)
3. Преподаватели курсов (Преподаватель)
4. Администратор сайта (Администратор)

Система должна предоставлять посетителю следующие возможности:

1) Просмотреть информацию о курсах
2) Просмотреть информацию об организации
3) Обратиться в службу поддержки
4) Авторизоваться
   a) Войти в аккаунт

b) Создать аккаунт

Система должна предоставлять студенту следующие возможности:

1) Купить курс
   a) Выбрать метод оплаты
      i) Выбрать оплату картой онлайн
      ii) Выбрать оплату платёжным сервисом
      iii) Оформить рассрочку на покупку
   b) Применить промо-код на скидку
2) Открыть доступный курс
   a) Перейти в модуль
      i) Открыть урок
         (1) Просмотреть обучающие видео урока
         (2) Сдать домашнюю работу
         (3) Написать курирующему преподавателю
3) Просмотреть свой профиль
   a) Редактировать личные данные
   b) Изменить пароль

Система должна предоставлять администратору следующие возможности:

1) Управлять курсами
   a) Добавить курс
   b) Удалить курс
   c) Изменить курс
      i) Изменить название курса
      ii) Добавить модуль
      iii) Удалить модуль
      iv) Изменить модуль
         (1) Изменить название модуля
         (2) Добавить урок
         (3) Удалить урок
         (4) Изменить урок

(a) Добавить видео

(b) Удалить видео

(c) Добавить домашнюю работу

(d) Удалить домашнюю работу

2) Управлять пользователями
   a) Добавить пользователя
   b) Удалить пользователя
   c) Редактировать информацию пользователя
3) Прочитать обращения из службы поддержки

Система должна предоставлять преподавателю следующие возможности:

1) Проверить домашнее задание
   a) Вернуть домашнее задание
   b) Принять домашнее задание
   c) Оставить комментарий
2) Просмотреть свой профиль
   a) Редактировать личные данные
   b) Изменить пароль

1.2 Постановка задачи анализа активностей пользователя

Перед нами стоит следующая задача – на основании проведённого исследования данных сформировать аналитическую сводку для разработчиков образовательных программ с целью улучшения качества контента и выявления студентов с проблемами в успеваемости по учебным программам. В начале должно быть сформулировано описание всех курсов и рассчитанных метрик. Следующим этапом необходимо будет рассчитать активность пользователей, чтобы понять, на какое время приходится наибольшая нагрузка на преподавателей, а значит на каких образовательных программах в будущем потребуется больше сотрудников. Затем идет блок из двух пунктов по анализу качества контента курсов, где необходимо выявить проблемные модули, которые, возможно, требуют доработки. Также стоит задача выявить потенциальную сезонность. Будет разработана метрика

успеваемости студентов для нахождения тех, кто значительно хуже справляются с прохождением курса. Также будет проведён разведочный анализ данных с целью уменьшения количества исследуемых атрибутов и нахождения зависимостей между переменными. В последнюю очередь будет испробовано несколько моделей машинного обучения и оценены результаты их работы для выявления лучшего подхода.

1.3 Обзор и исследование языка моделирования UML для разработки информационных систем

Строительные блоки UML

Словарь языка UML включает три вида строительных блоков: сущности, отношения, диаграммы.

Сущности

Сущности – это абстракции, являющиеся основными элементами модели. Они являются наиболее важными строительными блоками UML. В UML имеется четыре типа сущностей: структурные, поведенческие, группирующие, аннотационные.

Структурные сущности

Структурные сущности определяют статическую часть модели. Они представляют физические и концептуальные элементы. Ниже приведены краткие описания структурных сущностей.

Класс — это описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой (см. рис. 1.3.1).

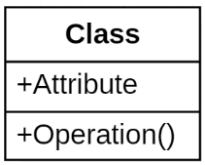

Рис. 1.3.1. Класс

Интерфейс — это совокупность операций, которые определяют сервис (набор услуг), предоставляемый классом или компонентом (см. рис. 1.3.2). Таким образом, интерфейс описывает видимое извне поведение элемента.

Рис. 1.3.2. Интерфейс

Кооперация — определяет взаимодействие; она представляет собой совокупность ролей и других элементов, которые, работая совместно, производят некоторый кооперативный эффект, не сводящийся к простой сумме слагаемых (см. рис. 1.3.3.)

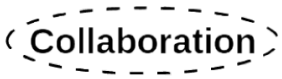

Рис. 1.3.3. Кооперация

Вариант использования — это описание последовательности выполняемых системой действий, которая производит наблюдаемый результат, значимый для какого-то определенного актера (см. рис. 1.3.4).

Рис. 1.3.4. Вариант использования

Компонент — это физическая заменяемая часть системы, которая соответствует некоторому набору интерфейсов и обеспечивает его реализацию (см. рис. 1.3.5). Компонент, как правило, представляет собой физическую упаковку логических элементов, таких как классы, интерфейсы и кооперации.

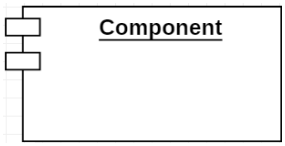

Рис. 1.3.5. Компонент

Узел — это элемент реальной (физической) системы, который существует во время функционирования программного комплекса и представляет собой

вычислительный ресурс, обычно обладающий как минимум некоторым объемом памяти, а часто еще и способностью обработки (см. рис. 1.3.6).

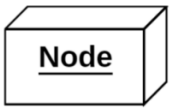

Рис. 1.3.6. Узел

Поведенческие сущности

Поведенческие сущности состоят из динамических частей моделей UML. Это глаголы языка: они описывают поведение модели во времени и пространстве. Существует всего два основных типа поведенческих сущностей.

Взаимодействие — это поведение, суть которого заключается в обмене сообщениями (Messages) между объектами в рамках конкретного контекста для достижения определенной цели. С помощью взаимодействия можно описать как отдельную операцию, так и поведение совокупности объектов. Взаимодействие предполагает ряд других элементов, таких как сообщения, последовательности действий (поведение, инициированное сообщением) и связи (между объектами). Графически сообщения изображаются в виде стрелки, над которой пишется имя соответствующей операции, как показано на рис. 1.3.7.

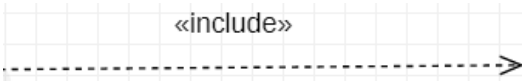

Рис. 1.3.7. Взаимодействие

Состояние — это некое положение в жизни объекта, при котором он удовлетворяет определенному условию, выполняет некоторое действие или ожидает события (см. рис. 1.3.8). Состояние объекта можно описать с помощью значений одного или нескольких атрибутов класса [5].

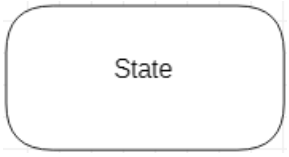

Рис. 1.3.8. Состояние

Группирующие сущности

Группирующие сущности являются организующими частями модели UML. Это блоки, на которые можно разложить модель. Есть только одна первичная группирующая сущность, а именно пакет.

Пакеты — представляют собой универсальный механизм организации элементов в группы. В пакет можно поместить структурные, поведенческие и даже другие группирующие сущности (см. рис. 1.3.9). [6] В отличие от компонентов, существующих во время работы программы, пакеты носят чисто концептуальный характер, то есть существуют только во время разработки.

Рис. 1.3.9. Пакет

Аннотационные сущности

Аннотационные сущности могут быть определены как механизм для сбора замечаний, описаний и комментариев элементов модели UML. Имеется только один базовый тип аннотационных элементов – примечание.

Примечание — это просто символ для изображения комментариев или ограничений, присоединенных к элементу или группе элементов (рис. 1.3.10).

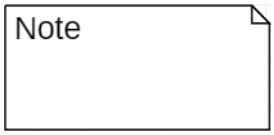

Рис. 1.3.10. Примечание

Отношения

Отношения — это еще один важнейший строительный блок UML. Он показывает, как элементы связаны друг с другом, и эта связь описывает функциональность приложения.

Есть четыре вида доступных отношений: зависимость, ассоциация, обобщение, реализация.

Зависимость

Зависимость — это семантическое отношение между двумя сущностями, при котором изменение одной из них, независимой, может повлиять на семантику другой, зависимой. Графически зависимость изображается в виде прямой пунктирной линии, часто со стрелкой, которая может содержать метку (см. рис. 1.3.11).

Рис. 1.3.11. Зависимость

Ассоциация

Ассоциация — структурное отношение, являющееся одним из фундаментальных понятий в языке UML, которое описывает совокупность связей, где под связью понимается соединение между объектами. Ассоциация может использоваться на различных канонических диаграммах при построении визуальных моделей [7]. Разновидностью ассоциации является агрегирование (Aggregation) - так называют структурное отношение между целым и его частями. Графически ассоциация изображается в виде прямой линии (иногда завершающейся стрелкой или содержащей метку), рядом с которой могут присутствовать дополнительные обозначения, на пример кратность и имена ролей. На рис. 1.3.12 показан пример отношений этого типа.

Рис. 1.3.12. Ассоциация

Обобщение

Обобщение — это отношение "специализация/обобщение", при котором объект специализированного элемента (потомок) может быть подставлен вместо объекта обобщенного элемента (родителя или предка) (см. рис. 1.3.13).

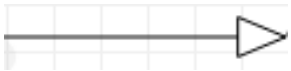

Рис. 1.3.13. Обобщение

Реализация

Реализация — это семантическое отношение между классификаторами, при котором один классификатор определяет "контракт", а другой гарантирует его

выполнение. Отношения реализации встречаются в двух случаях: во-первых, между интерфейсами и реализующими их классами или компонентами, а во-вторых, между прецедентами и реализующими их кооперациями [8] (см. рис. 1.3.14).

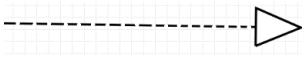

Рис. 1.3.14. Реализация

Отношения связывают различные сущности; диаграммы группируют представляющие интерес совокупности сущностей.

Диаграммы

Диаграмма — это графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями). Диаграммы рисуют для визуализации системы с разных точек зрения. В UML выделяют восемь типов диаграмм, которые разделяются на два глобальных типа: Структурные диаграммы, Диаграммы поведения.

Структурные диаграммы

- **Классов** — позволяют описать модель классов и их отношения при помощи графической системы обозначений [9]. При моделировании объектно-ориентированных систем этот тип диаграмм используют чаще всего.
- **Объектов** — представлены объекты и отношения между ними. Они являются статическими "фотографиями" экземпляров сущностей, показанных на диаграммах классов.
- **Компонентов** — представлена организация совокупности компонентов и существующие между ними зависимости. Они могут быть соотнесены с диаграммами классов, так как компонент обычно отображается на один или несколько классов, интерфейсов или кооперацией.
- **Размещения** — представлена конфигурация обрабатывающих узлов системы и размещенных в них компонентов. Диаграммы поведения
- **Последовательности** — отражают временную упорядоченность сообщений. С помощью них можно проиллюстрировать взаимодействие исполнителя с системой и операции, выполнение которых при этом инициализируется [10].

- **Коммуникации** — отражает структурную организацию обменивающихся сообщениями объектов.
- **Автомата / состояний** — на диаграммах состояний представлен автомат, включающий в себя состояния, переходы, события и виды действий.
- **Деятельности** — это частный случай диаграммы состояний; на ней представлены переходы потока управления от одной деятельности к другой внутри системы.

Разработка диаграмм для информационной системы онлайн-образования

Как уже описывалось в техническом задании, в системе были выделены четыре вида пользователей: Посетитель, Студент, Преподаватель, Администратор. Для каждого пользователя составлены по одной диаграмме вариантов использования (рис. 1.3.15-1.3.18), по одной диаграмме классов (рис. 1.3.19-1.3.22) и по одной диаграмме последовательности действий для наиболее важных вариантов использования (рис. 1.3.23-1.3.25).

Диаграммы вариантов использования

В представленной системе пользователь Посетитель является родительской сущностью пользователей Преподаватель и Студент, передавая им свои варианты использования, однако с той лишь разницей, что Преподаватель лишён возможности самостоятельно зарегистрироваться, и нуждается в создании своего аккаунта со стороны администратора.

Посетитель имеет возможность совершать следующие действия:

1. Просмотреть информацию о курсе – означает, что любой пользователь системы может ознакомиться с информацией об актуальных курсах подготовки: название курса, описание, модули, количество часов лекций, предполагаемое время обучения и цена.
2. Просмотреть информацию об организации – означает, что любой пользователь системы может ознакомиться с информацией о сайте (об организации) на базе которой проходит обучение: название, описание, физический адрес, контакты и юридическая информация.

3. Обратиться в службу поддержки – это возможность любого пользователя, даже незарегистрированного, написать обращение в службу поддержки, при этом оставив адрес своей электронной почты для ответа.

4. Авторизоваться – это возможность для неавторизованного Посетителя войти в систему, перейдя в статус Студента или Преподавателя, или же зарегистрировать аккаунт самостоятельно, тем самым став Студентом.

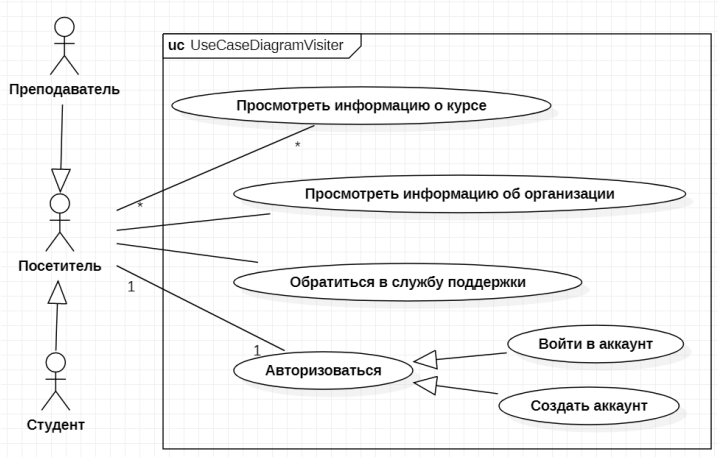

Рис. 1.3.15. Диаграмма вариантов использования для посетителя

Студент – это дочерний класс Посетителя. Студентом становится Посетитель сайта, который прошёл процесс авторизации.

Студент имеет возможность совершать следующие действия:

1. Купить курс – это приобретение доступа к выбранному курсу путём оплаты такового. Для совершение этого действия следует прежде всего выбрать метод оплаты: банковская карта, платёжный сервис (PayPal, Qiwi, Yandex.Деньги и т.п.) или же оформление рассрочки на покупку через банк партнёра, что по является беспроцентным кредитом.

2. Открыть доступный курс – это процесс прохождения уже приобретённого курса или множества курсов. Процесс включает в себя переходы в меньшие единицы, являющиеся составными частями курса (модули, уроки). Каждый урок включает в себя обучающее видео и/или домашнее задание, которое требуется сдавать через систему.

3. Просмотреть свой профиль – это действие подразумевает просмотр данных, предоставленных пользователем при регистрации, а также их изменение (личные данные, пароль).

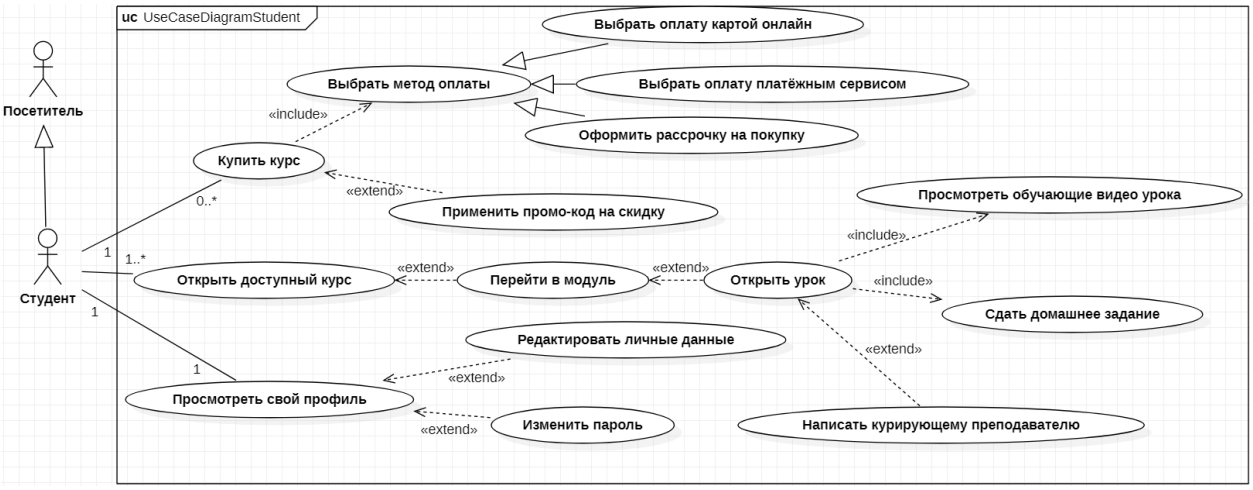

Рис. 1.3.16. Диаграмма вариантов использования для студента

В представленной диаграмме Администратор является родительским классом пользователя Преподаватель и по усмотрению может предоставлять ему права на некоторые действия, таки как управление курсами, управление пользователями и прочтение обращений в службу поддержки.

Администратор имеет возможность совершать следующие действия:

1. Управлять курсами – это возможность пользователя добавлять новые, а также изменять или удалять существующие курсы. В ходе изменения курса существует возможность изменить его название или состав, добавив, изменив или удалив модуль. Изменяя модуль, Администратор также может изменить его название или состав, добавив, изменив или удалив урок. В ходе редактирования содержимого урока пользователь имеет возможность добавлять и удалять обучающее видео или домашнюю работу.

2. Управлять пользователями – под этим действием подразумевается администрирование зарегистрированными пользователями посредством информационной системы. В ходе данного управления пользователь может быть добавлен или удалён администратором вручную, а также информация о нём изменена, если речь идёт о пользователе Преподаватель.

3. Прочитать обращения из службы поддержки – это действие направлено на проверку сообщений, которые могут поступить от любого пользователя системы. В данном случае администратор должен разобраться в ситуации и отправить ответ на ту электронную почту, которая была указана в обращении.

Рис. 1.3.17. Диаграмма вариантов использования для администратора

Преподаватель является дочерним пользователем Администратора и Посетителя, тем самым унаследовав возможности обоих актёров, за исключением возможности самостоятельно создавать аккаунт.

Преподаватель имеет возможность совершать следующие действия:

1. Проверить домашнее задание – это возможность пользователя просмотреть то, что в качестве домашнего задания ему прислал Студент, и после этого принять решение о том, засчитывается работа или нет. В качестве дополнения, за преподавателем остаётся возможность оставить свои комментарии, тем самым похвалив, направив в нужном направлении, дав совет или ответив на прямой вопрос Студента.

2. Просмотреть свой профиль – это действие подразумевает просмотр данных, введённых Администратором при регистрации пользователя в системе, а также их изменение (личные данные, пароль).

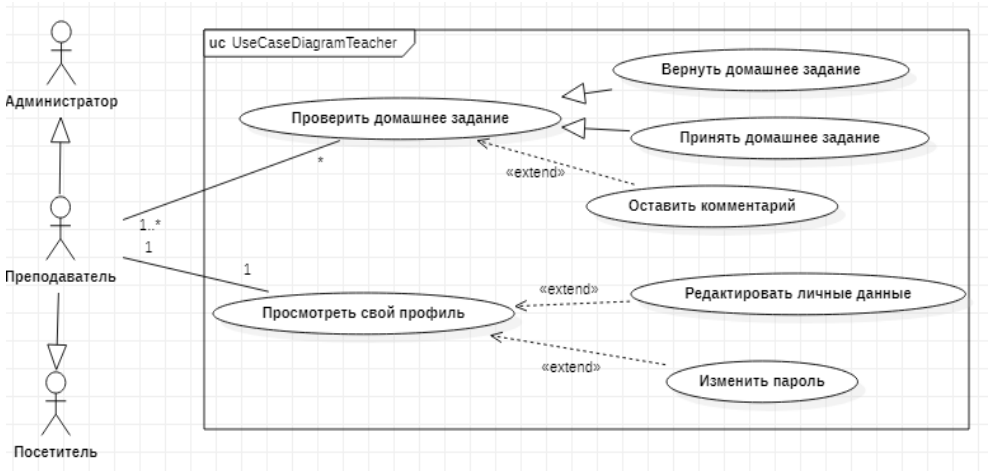

Рис. 1.3.18. Диаграмма вариантов использования для преподавателя

Диаграммы классов

На основе диаграмм вариантов использования были разработаны диаграммы классов для каждого типа пользователей (рис. 1.3.19-1.3.22).

IПосетитель представляет собой пользовательский интерфейс, благодаря которому можно просмотреть информацию об имеющихся курсах или об организации (онлайн-портале), авторизоваться в системе, обратиться в службу поддержки, описав возникшую проблему и указав электронную почту для обратной связи.

При авторизации происходит переход к интерфейсу IАвторизация, в котором возможно одно из двух действий: вход в существующий аккаунт с указанием электронной почты и пароля для идентификации, создание аккаунта с введением таких регистрационных данных, как имя, пол, город, день рождения, электронная почта и пароль.

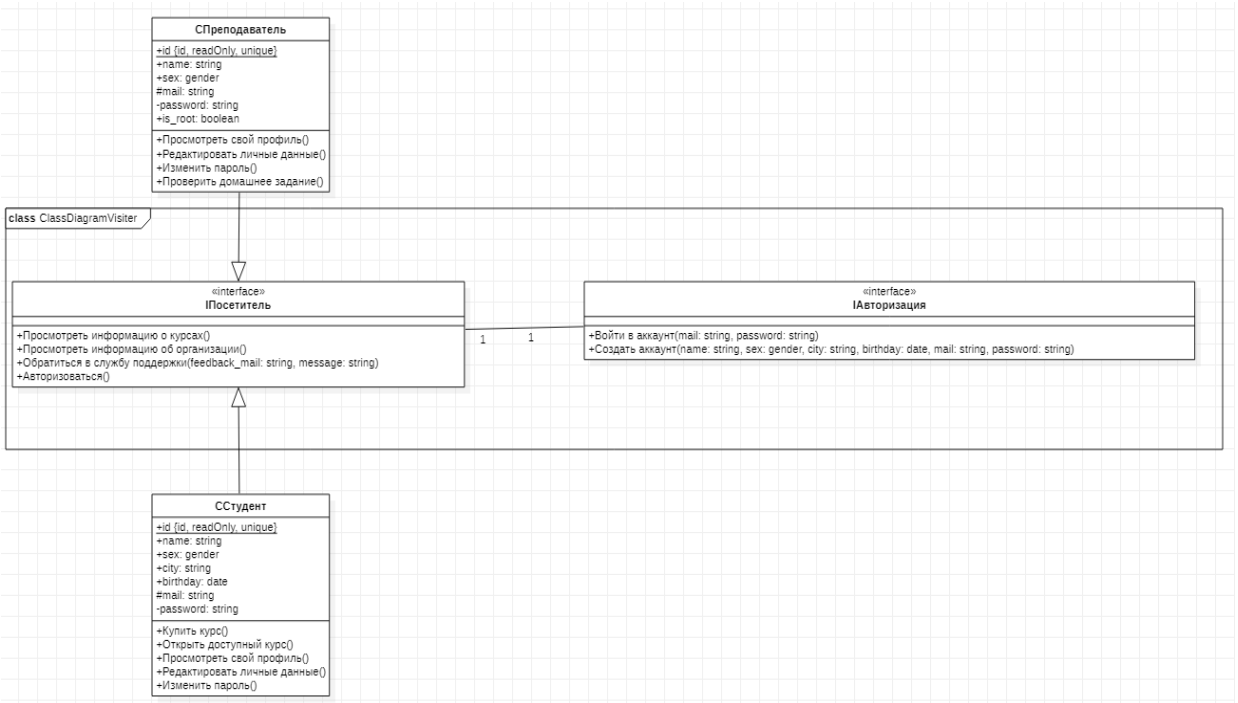

Рис. 1.3.19. Диаграмма классов для посетителя

ССтудент представляет собой класс, каждый экземпляр которого имеет следующие атрибуты: уникальный идентификатор id, имя name типа string, пол sex собственного типа gender, город проживания city типа string, день рождения birthday типа date, электронная почта mail типа string, пароль password типа string. Каждый экземпляр класса имеет следующий набор операций: «Купить курс», «Открыть доступный курс», «Просмотреть свой профиль», «Редактировать личные данные», «Изменить пароль».

При вызове операции «Открыть доступный курс» происходит переход к экземпляру класса СКурс, который имеет следующие атрибуты: уникальный идентификатор course_id, название курса title типа string, сфера, к которой относится курс, field типа string. Аналогичный переход происходит при вызове операции «Купить курс», однако с тем отличием, что при покупке курса происходит обращение к интерфейсу IОплата, в котором можно оплатить выбранный курс, выбрать способ оплаты и применить промо-код на скидку. При выборе способа оплаты вызывается соответствующий интерфейс со следующими вариантами оплаты: «Выбрать оплату картой онлайн», «Выбрать оплату платёжным сервисом», «Оформить рассрочку на покупку».

Если же курс уже приобретён, то появляется возможность выполнить операцию «Перейти в модуль», тем самым перейдя к экземпляру класса СМодуль, который имеет следующие атрибуты: номер модуля module_number типа int, название модуля module_title типа string. Каждый экземпляр класса имеет единственную операцию «Открыть урок», при котором осуществляется переход к экземпляру класса СУрок.

СУрок представляет собой класс, каждый экземпляр которого имеет следующие атрибуты: номер урока lesson_number типа int, название урока lesson_title типа string, токен урока lesson_token типа string, атрибут наличия обучающего видео is_video типа boolean, атрибут наличия домашней работы is_homework типа boolean. Каждый экземпляр класса имеет следующий набор операций: «Просмотреть обучающие видео урока», «Сдать домашнюю работу», «Написать курирующему преподавателю».

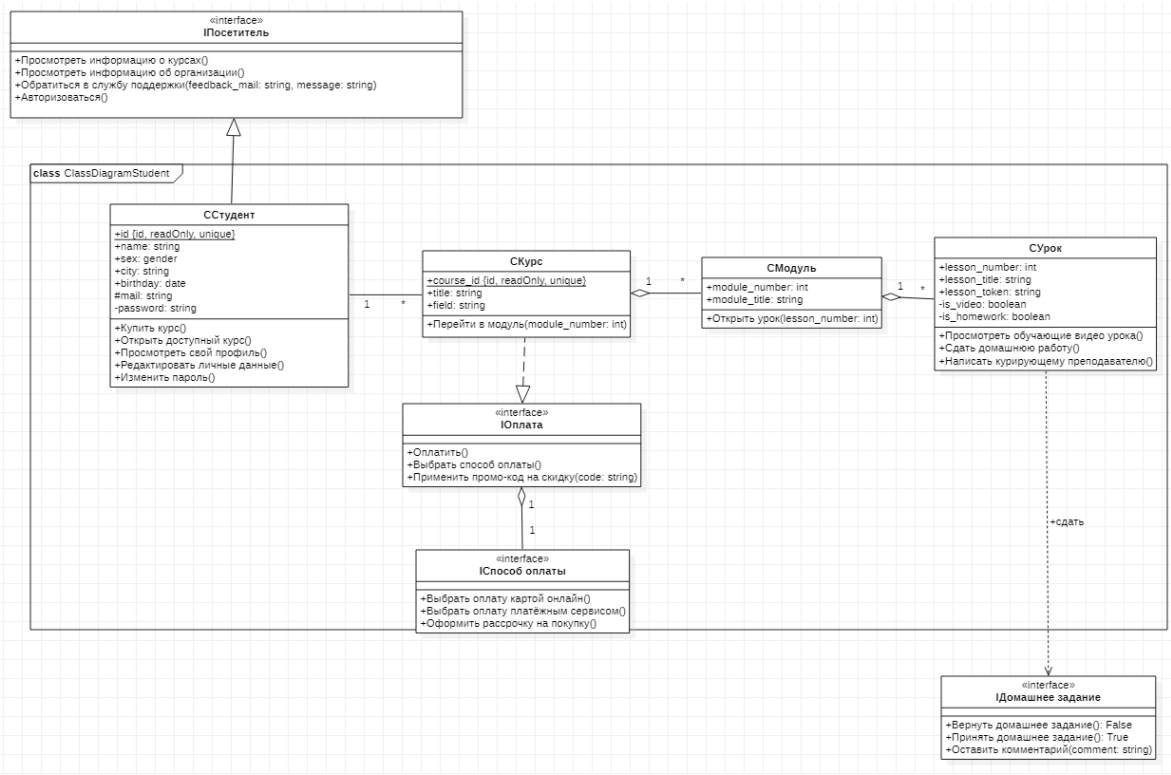

Рис. 1.3.20. Диаграмма классов для студента

САдминистратор представляет собой класс, каждый экземпляр которого имеет следующие атрибуты: уникальный идентификатор id, имя name типа string, электронная почта mail типа string, пароль password типа string, константный атрибут

is_root типа boolean равный True. Каждый экземпляр класса имеет следующий набор операций при условии is_root=True, что означает наличие прав доступа администратора: «Добавить курс», «Удалить курс», «Изменить курс», «Добавить пользователя», «Удалить пользователя», «Редактировать информацию пользователя». Операция «Прочитать обращения из службы поддержки» доступна и без прав администратора, так как она может быть делегирована классу СПреподаватель.

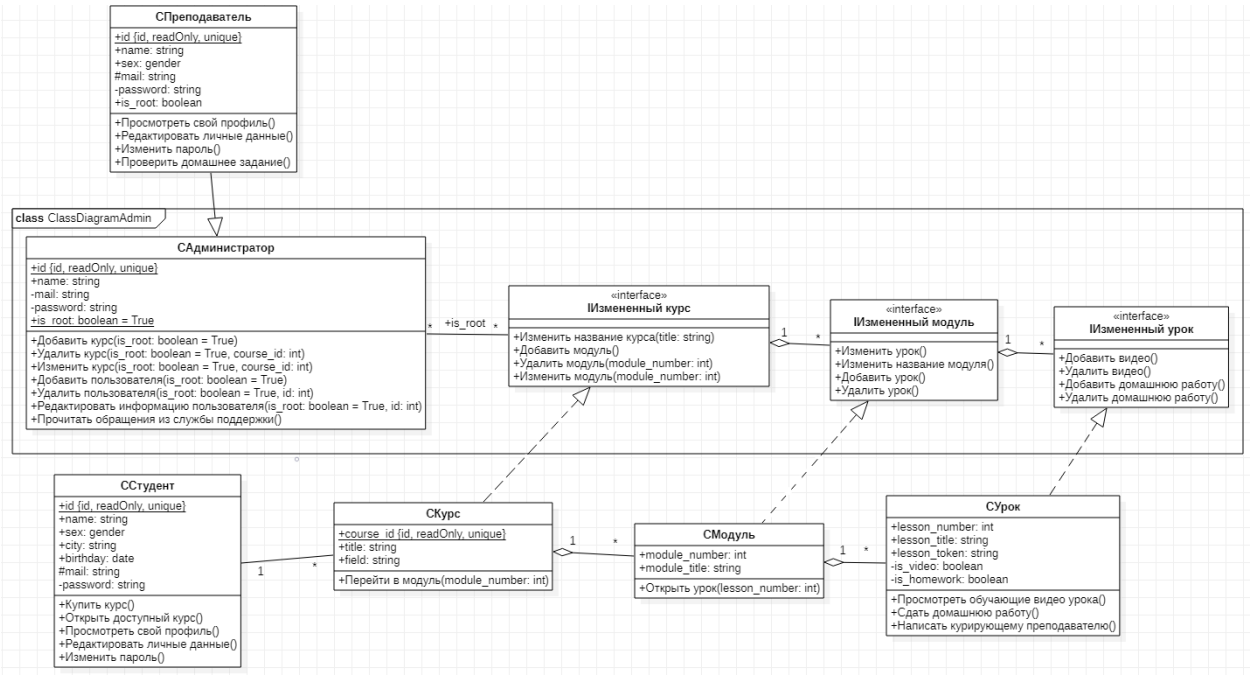

Рис. 1.3.21. Диаграмма классов для администратора

СПреподаватель – это дочерний класс одновременно класса САдминистратор и интерфейса IПосетитель, каждый экземпляр которого имеет следующие атрибуты: уникальный идентификатор id, имя name типа string, пол sex собственного типа gender, электронная почта mail типа string, пароль password типа string, атрибут is_root типа boolean. Так как данный класс является дочерним от САдминистратор, то каждый экземпляр класса также имеет следующий набор операций при условии is_root=True, что означает наличие прав доступа администратора: «Добавить курс», «Удалить курс», «Изменить курс», «Добавить пользователя», «Удалить пользователя», «Редактировать информацию пользователя». Каждый экземпляр класса имеет следующий набор операций при любом значении параметра is_root: «Просмотреть свой профиль», «Редактировать личные данные», «Изменить пароль»,

«Проверить домашнее задание». При вызове операции «Проверить домашнее задание» происходит переход к интерфейсу IДомашнее задание, которое создаётся по ходу операции «Сдать домашнюю работу» из класса СУрок.

Интерфейс IДомашнее задание имеет следующий набор операций: «Вернуть домашнее задание» возвращающее значение False, «Принять домашнее задание» возвращающее значение True, «Оставить комментарий».

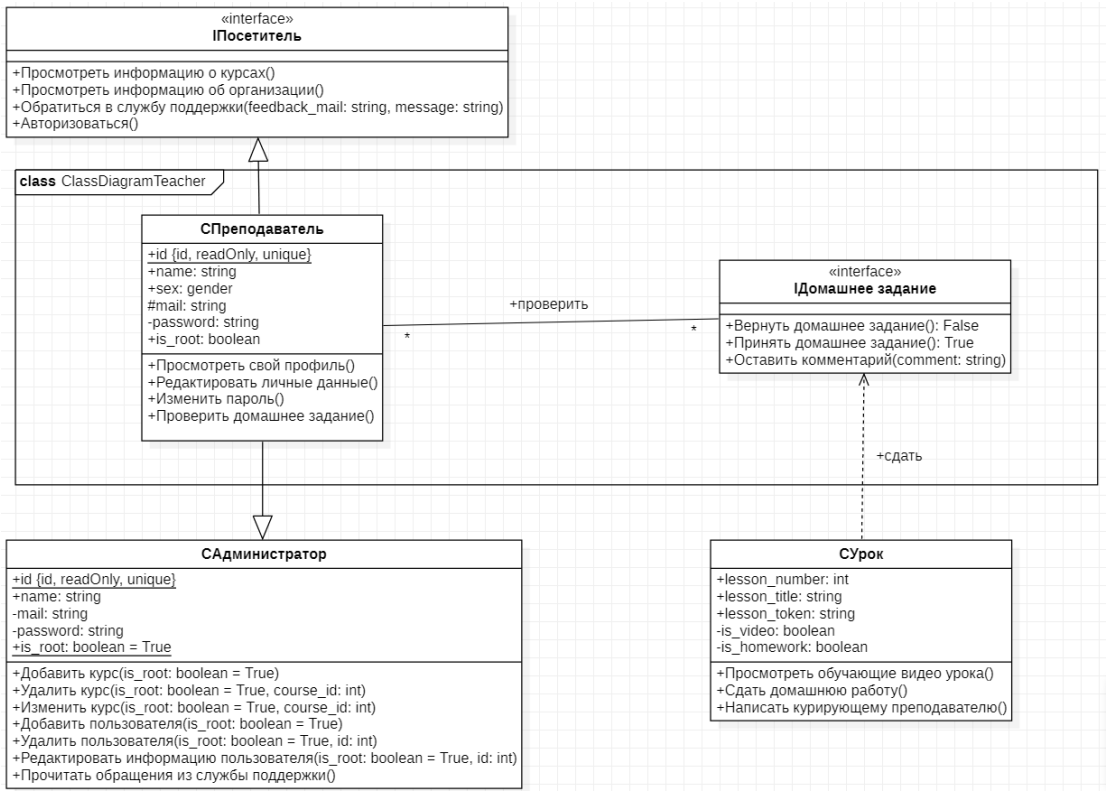

Рис. 1.3.22. Диаграмма классов для преподавателя

Диаграммы последовательности

На основе диаграмм вариантов использования были разработаны диаграммы последовательности действий для некоторых вариантов использования (рис. 1.3.23-1.3.25).

При авторизации посетителя сайта через интерфейс IПосетитель происходит переход к интерфейсу IАвторизация. Если у пользователя нет зарегистрированного аккаунта, то он проходит процесс создания аккаунта, при котором инициализируется экземпляр класса ССтудент. После этого пользователь может войти в систему посредством интерфейса IАвторизация.

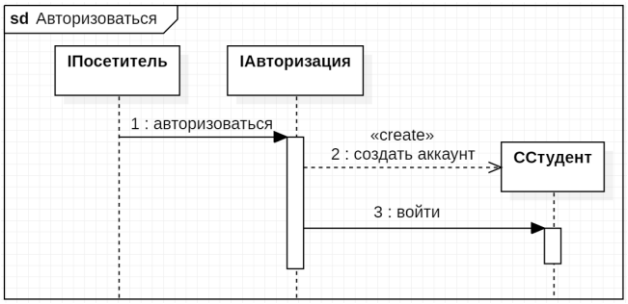

Рис. 1.3.23. Диаграмма последовательности действий для варианта использования «Авторизоваться»

Для проверки домашнего задания со стороны преподавателя класса СПреподаватель в первую очередь должна быть сдана домашняя работа определённого экземпляра класса СУрок, после чего создаётся интерфейс IДомашнее задание. После этого СПреподаватель может проверить IДомашнее задание и сделать одно из двух действий: вернуть задание на доработку, принять домашнее задание. Также вместе с любым решением, пользователь может оставить комментарий по своему усмотрению.

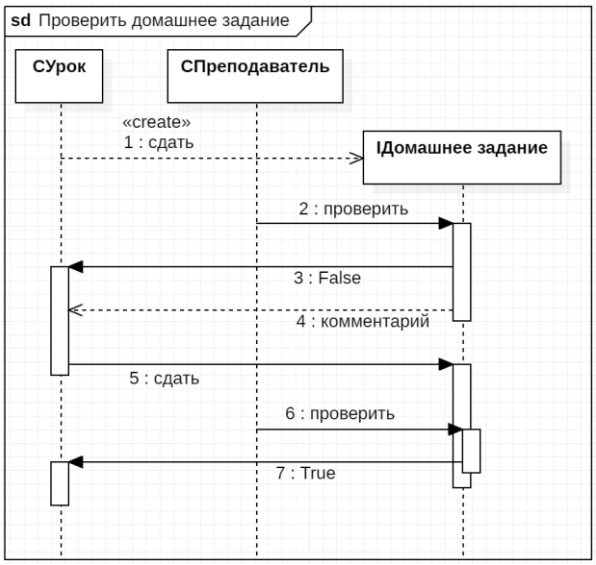

Рис. 1.3.24. Диаграмма последовательности действий для варианта использования «Проверить домашнее задание»

При решении изменить какой-либо курс САдминистратор совершает переход к интерфейсу IИзмененный курс, где следует выбрать что именно изменить. Если было принято решение об изменении названия курса, то в экземпляре класса СКурс задаётся новое значение атрибута title.

Рис. 1.3.25. Диаграмма последовательности действий для варианта использования «Изменить курс»

Архитектура

Архитектуру системы было принято изобразить на основе диаграммы компонентов (Рис. 1.3.26). Модули видео и домашнего задания вынесены как увеличивающие функционал системы. В то же время система не зависит от них, и сами эти модули самодостаточны. Модуль авторизации вынесен в отдельную часть, поскольку он не реализует дополнительный функционал системы, а является одной из ее частей.

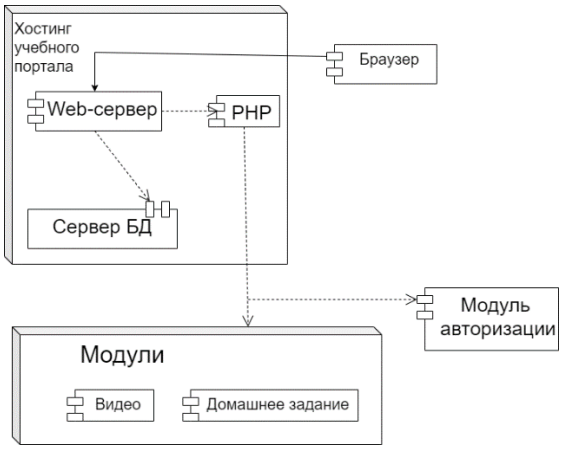

Рис. 1.3.26. Архитектура системы

2 Современные методы машинного обучения и их применение для решения поставленной задачи

2.1 Обзор современных методов машинного обучения

Рассмотрим основные типы машинного обучения. Согласно классическому разделению, выделяют три типа [11]:

- обучение с учителем,

- обучение без учителя,
- обучение с подкреплением.

Однако существуют ещё гибридные типы машинного обучения:

- обучение с частичным привлечением учителя,
- глубокое обучение.

Обучение с учителем

Термин обучение «с учителем» означает, что на первом этапе используются заранее размеченные данные, в которых нужные выходные сигналы (метки) уже известны. Параметры модели и размер выборки можно менять до тех пор, пока полученный результат не будет удовлетворительным. Для проверки статистических параметров модели можно подать на ввод данные, с которыми система ещё не работала, а после этого сравнить предсказанные данные с настоящими.

Обучение с учителем широко применяется в задачах прогнозирования и классификации. Такой вид обучения можно использовать, к примеру, в финансовых организациях для определения рисков вложений или выдачи кредитов на основе финансового поведения клиента.

Обучение без учителя

В ходе обучения без учителя система выявляет скрытые свойства и корреляционные закономерности между атрибутами. На основе их статистических свойств можно объединять данные в кластеры на основании одних только статистических свойств.

Примером использования обучения без учителя является алгоритм кластеризации, при котором определяются вероятностные отношения связи между элементами данных, а уже после этого обнаруживаются взаимосвязи между исследуемыми объектами.

В качестве одного из примеров использования обучения без учителя является алгоритм кластеризации, при котором определяются вероятностные отношения связи между элементами данных, а уже после этого обнаруживаются взаимосвязи между исследуемыми объектами.

Обучение с частичным привлечением учителя

Это сочетанием двух подходов: обучения с учителем и без. После разметки небольшой части данных, модели машинного обучения предоставляется возможность понять, каким именно образом следует проводить кластеризацию оставшегося датафрейма.

Задание размеченных данных для задачи обучения часто требует наличия квалифицированного человека в качестве эксперта или проведения физического эксперимента. Поэтому временные и финансовые затраты на разметку данных могут превратить процесс обучения с использованием лишь размеченных данных в невыполнимую задачу, в то время как процесс задания неразмеченных данных не является очень затратным. В таких случаях обучение с частичным привлечением учителя может иметь большое значение на практике. Такой вид обучения также представляет интерес в сфере машинного обучения и как модель для человеческого обучения.

Обучение с подкреплением

При обучении с подкреплением система развивается сама, не основываясь на подготовленных данных. Её дальнейшая эволюция зависит от взаимодействия с окружением, а при правильном решении задачи модели полагается условное вознаграждение. Таким образом, накопление вознаграждений формирует поведение модели в правильном направлении.

На данный момент проводится большое количество экспериментов по обучению с подкреплением в сфере видеоигр. Системы обучаются с нуля и не совершают никаких действий, однако, пробуя предпринять что-то новое, они постепенно начинают понимать, чего от них хотят. С каждым достижением поставленной задачи системы справляются всё успешнее, а при изменении изначальных условий среды быстрее адаптируются.

Как правило, этот тип машинного обучения применяют в тех случаях, когда методы обучения с учителем и без показали низкую эффективность.

Глубокое обучение

При использовании глубокого обучения в качестве основы выступают нейронные сети, для итерационного уточнения характеристик имеющихся данных. Обучать такие системы можно как с подкреплением, так и без учителя.

Глубокие нейронные сети часто применяются, например, для ускорения процесса скрининга больших объемов данных в ходе поиска новых лекарственных средств. Нейросети такого рода способны обрабатывать огромное количество изображений за короткий срок и извлечь большее количество признаков, которые модель в конечном счете запоминает.

Глубокие нейронные сети находят активное применение в современном мире. К примеру, известны случаи применения глубокого обучения для подбора молекулярных формул новых лекарств. Однако есть и негативная сторона использования такого вида систем. Не так давно глубокие нейронные сети стали использоваться для создания «Deepfake» видео, где лица известных людей подставляются к телам специально нанятых актёров или просто к участникам видеоролика из интернета, тем самым дискредитируя личность. [12]

Методы машинного обучения

Часто используемые методы обучения с учителем:

- K-ближайших соседей
- Линейная регрессия
- Нейронные сети
- Метод опорных векторов
- Логистическая регрессия
- Деревья решений и случайные леса

K-ближайших соседей (k-NN)

В ходе использования этого метода модель делает прогноз на определение класса объекта из тестовой выборки, проводя анализ *k объектов из* обучающей выборки, которые по расстоянию расположены ближе всего к тестовому объекту.

В методе *k* ближайших соседей выбирается тот класс, который получает большинство «голосов» от близлежащих объектов.

Выбор нечетного значения *k* в алгоритме *k*-NN предотвращает «ничьи» и гарантирует, что количество голосов никогда не будет равным.[13]

Линейная регрессия

При помощи метода *простая линейная регрессия*, можно построить прогнозы, выявляющие линейные отношения между данными. Для заданной коллекции значений, представляющих *независимую переменную* и *зависимую переменную*, простая линейная регрессия описывает отношение между этими переменными прямой линией — *регрессионной прямой*.

Используя простую линейную регрессию, можно определить коэффициент наклона и точку пересечения прямой линии, которые наилучшим образом подходят к набору данных.

При использовании *множественной линейной регрессии*, для расчёта берутся все имеющиеся числовые признаки, чтобы построить более сложные прогнозы, чем при использовании только одного признака или подмножества признаков.

Множественная линейная регрессия выдает для каждого признака отдельный коэффициент и отдельную точку пересечения. [14]

Нейронные сети

Искусственная нейронная сеть (или просто нейронная сеть) представляет собой сложную программную конструкцию, при построении которой программист закладывает те же принципы работы, по каким работает мозг человека.

Нейронные сети пытаются имитировать структуру нейронов головного мозга: каждый искусственный нейрон соединяется с несколькими другими нейронами. Нейросети имеют многослойную структуру: нейроны, расположенные на первом слое, передают данные нескольким нейронам на втором и т. д. В конце данные передаются в выходной слой, где нейронная сеть выдает своё предположение о том, каким образом решить поставленную задачу, классифицировать объект и т. п.

Нейросети применяются для решения целого ряда задач в различных отраслях. В здравоохранении их используют для анализа медицинских снимков, чтобы ускорить диагностику и для поиска лекарств путём подбора молекулярной структуры вакцины. В телекоммуникационной отрасли и медиаиндустрии нейросети

применяются для машинного перевода и оказывания услуг виртуальных ассистентов. В финансовой сфере их используют для распознавания попыток мошенничества, управления инвестиционными портфелями и анализа риска. В розничной торговле — для уменьшения очередей в кассу и при идентификации покупателей для обслуживания. [12]

Метод опорных векторов

При решении задач классификации и регрессии с помощью метода опорных векторов – Support Vector Machines (SVM), ставится цель разработки алгоритмически эффективных методов построения оптимальной разделяющей гиперплоскости в пространстве признаков большой размерности. В данном случае, под оптимальностью понимается минимизация верхних оценок вероятности ошибки обобщения. [15]

SVM — это системы обучения, которые используют пространство гипотез линейных функций в многомерном пространстве признаков, обученных при помощи алгоритма обучения из теории оптимизации, который реализует предвзятость обучения, полученную из статистической теории обучения. Такая стратегия обучения, представляет собой эффективный метод, который за несколько лет с момента его появления в сфере машинного обучения показал лучшую эффективность по сравнению с большинством других систем в самых разных приложениях. [16]

Логистическая регрессия

В регрессионном анализе логистическая регрессия оценивает параметры логистической модели (коэффициенты в линейной комбинации). Формально в бинарной логистической регрессии есть одна бинарная зависимая переменная, закодированная индикаторной переменной, где два значения помечены как «0» и «1», а каждая из независимых переменных может быть бинарной переменной (два класса, закодированные как индикаторная переменная) или непрерывной переменной (любое действительное значение).

Логистическая регрессия может показать, какой из различных оцениваемых факторов имеет наибольшую связь с результатом, и обеспечивает меру величины

потенциального влияния. Она также имеет возможность «приспосабливаться» к вмешивающимся факторам, т. е. факторам, которые связаны как с другими предикторными переменными, так и с результатом, поэтому мера влияния интересующего предиктора не искажается эффектом вмешивающегося фактора.

Хотя логистическую регрессию можно использовать для оценки эпидемиологических ассоциаций, которые не отражают причину и следствие, в основном в медицине её используют для создания моделей для прогнозирования результатов лечения пациентов. В этом контексте термин «предикторы» используется для обозначения независимых факторов (переменных), для которых количественно определяется влияние, а термин «результат» используется для зависимой переменной, которую пытается предсказать модель логистической регрессии. [17]

Дерево решений

Метод дерева решений осуществляет классификацию объектов, отвечая на «вопросы» об их атрибутах, которые располагаются в узловых точках. В зависимости от ответа делается выбор в пользу одной из ветвей, и таким образом алгоритм продолжается до тех пор, пока не будет достигнут «лист» — окончательный ответ.

В список применений дерева решений входят платформы управления знаниями для клиентского обслуживания, прогнозированного назначения цен и планирования выпуска продукции.

В сфере страховых услуг дерево решений может выяснить, какие виды страховой продукции и премий лучше всего задействовать с учетом возможного риска. Используя информацию о местонахождении и сведения о страховых случаях с учетом погодных условий, система имеет возможность определять категории риска на основании поданных требований и затраченных сумм. Затем, используя модели, система будет оценивать новые заявления страховых случаев, классифицируя их по категории риска и возможному финансовому ущербу.

«Случайный лес»

Для того, чтобы одно дерево решений выдавало точные результаты, его нужно обучать, метод же случайного леса (random forest) использует «комитет» случайным образом созданных решающих деревьев с различными наборами атрибутов и предоставляет возможность им проголосовать за самый популярный класс.

Случайный лес — универсальный, быстро обучаемый механизм для обнаружения связей внутри набора данных. В качестве примера можно привести массовые рассылки спама, что создают неудобства не только пользователям, но и вызывают проблемы у провайдеров Интернета, которым из-за нежелательных писем приходится иметь дело с возросшей нагрузкой на серверы. Для борьбы с подобными проблемами были разработаны автоматические методы фильтрации спама, которые с помощью ансамбля решающих деревьев с высокой скоростью и большой эффективностью идентифицируют нежелательные письма.

Среди других применений — диагностика заболеваний путем анализа медицинской карты пациента, распознавание банковских мошенничеств, прогнозирование числа звонков в колл-центрах и прогнозирование вероятности прибыли и убытка при покупке определенных акций. [12]

2.2 Применение методов машинного обучения для анализа активностей пользователя

Для анализа активности пользователей, в основном используют модели обучения с учителем, так как в большинстве случаев имеется достаточно много данных о клиентах и меток их поведения или категорий. Самыми популярными являются следующие алгоритмы классификации: деревья решений, случайный лес, градиентный бустинг, метод k-ближайших соседей, логистическая регрессия. [18]

2.3. Аналитическое сравнение изученных методов и основные выводы

Для более простого восприятия информации о сравниваемых методах представлена в табл. 2.3, в которой отображаются плюсы и минусы тех или иных методов машинного обучения.

Таблица 2.3 – Преимущества и недостатки каждого изученного метода

Метод	Плюсы	Минусы
К-ближайших соседей	• Простая реализация; • Проработанная теоретическая база; • Адаптация под нужную задачу выбором метрики или ядра; • Интерпретируемость	• Недостаточная производительность в реальных задачах; • Трудность в наборе подходящих весов и определением, какие признаки необходимы для классификации; • Зависимость от выбранной метрики расстояния между примерами
Линейная регрессия	• Простая реализация и несложный расчет	• Невозможно уместить нелинейные данные
Нейронные сети	• Распараллеленная обработка данных предоставляет преимущества в виде высокой способности к обучению и эффективному распределению памяти; • Высокая надёжность; • Используется функция контентно-адресуемой памяти; • Низкая доля ошибок при классификации.	• Процесс обучения системы скрыт от пользователя, что негативно влияет на интерпретируемость результатов; • При настройке нейронных сетей необходимо указывать огромное число входных параметров; • Время обучения слишком велико и может даже не достичь цели обучения.

		• Метод расходует много вычислительных ресурсов, поэтому обучение может занимать большой промежуток времени.
Метод опорных векторов	• Есть возможность использовать действие нелинейных функций друг на друга; • Способен к решению многомерных функций с высокой сложностью; • Улучшает обобщение данных, за счёт чего отпадает необходимость использовать их все.	• При крайне большом количестве обрабатываемых записей эффективность построения моделей начинает снижаться; • Большая чувствительность метода к пропущенным значениям в данных; • Отсутствие универсального метода нахождения функции ядра классификатора делает затруднительным процесс поиска решения.
Логистическая регрессия	• Позволяет использовать меньшие ресурсы хранения при классификации данных за счёт сокращённого необходимого объёма вычислительной мощности.	• Неэффективен в решении многомерных функций с высокой сложностью; • Одновременно есть возможность выполнять лишь две задачи классификации ;

	При этом скорость остаётся крайне высокой. • За счёт своей простоты внедрения в рабочие процессы широко используется на массовых предприятиях.	• Для нелинейных функций требуется преобразование.
Деревья решений	• Высокая производительность обучения и прогнозирования; • Можно легко визуализировать и интерпретировать;	• Алгоритм игнорирует корреляционные связи между данными; • Склонность к переобучению; • Для данных с несогласованными размерами выборки в каждой категории в дереве решений результат получения информации смещен в сторону тех характеристик с большим количеством значений. • Если в выборках данные по каждой категории не согласованы между собой по размерам, то предпочтение алгоритма будет отдаваться характеристикам с

		превосходящим количеством значений.
Случайный лес	• Он преодолевает проблему переоснащения путем усреднения или объединения результатов различных деревьев решений. • Случайные леса хорошо работают с большим количеством элементов данных, чем одно дерево решений. • Случайный лес имеет меньшую дисперсию, чем одно дерево решений. • Случайные леса очень гибки и обладают очень высокой точностью. • Масштабирование данных не требует в алгоритме случайного леса. Он сохраняет хорошую точность даже после предоставления данных без масштабирования. • Алгоритмы Random Forest поддерживают хорошую точность даже при отсутствии значительной части данных.	• Сложность является основным недостатком алгоритмов случайного леса. • Построение Случайных лесов намного сложнее и отнимает больше времени, чем деревья решений. • Для реализации алгоритма Random Forest требуется больше вычислительных ресурсов. • Это менее интуитивно понятно в случае, когда у нас есть большая коллекция деревьев решений. • Процесс прогнозирования с использованием случайных лесов очень трудоемкий по сравнению с другими алгоритмами.

К-средних	• Алгоритм прост и удобен в реализации; • За счёт низкой сложности алгоритм имеет высокую эффективность и масштабируемость относительно данных. • Алгоритм пытается найти k разделов, которые минимизируют значение квадратичной функции ошибок. Когда кластеры являются плотными, сферическими или комковатыми, и разница между кластерами очевидна, эффект кластеризации лучше. • В ходе использования алгоритма ищутся k разделов, при которых минимизируется квадратичная функция ошибки. Чем очевиднее принимаемые кластерами фигуры и их границы, тем выше эффект кластеризации.	• Может сходиться к локальному минимуму, более медленная сходимость на крупномасштабных данных • Не все данные по своим характеристикам могут подойти; • Сложно сразу подобрать правильное значение K; • В зависимости от начального значения центра кластера разные начальные значения могут привести к разным результатам кластеризации; • Результат кластеризации сильно зависит от выбранного центра кластера; • Метод не следует использовать для выявление скоплений без выпуклых форм или имеющих разные относительно друг друга размеры.

В случае, если нет заранее заданного плана с выбранными методами, то, руководствуясь представленной выше информацией, можно следовать следующему алгоритму действий:

- В первую очередь следует опробовать логистическую регрессию. В случае, если модель имеет неудовлетворительные статистические показатели, то их можно использовать для сравнения с моделями на основе других алгоритмов;
- Далее следует построить дерево решений или даже случайный лес, если хватает вычислительной мощности или времени. При улучшении производительности можно остановиться и на этом варианте, однако алгоритм можно использовать для очистки данных от лишних переменных и продолжить поиск подходящего метода;
- В случае, когда имеется запас времени и вычислительной мощности, а также необходимо построить модель на основе большого количества данных и функций, следует использовать метод опорных векторов (SVM).

3 Разработка методов анализа активностей пользователя

Для решения задачи необходимо определить медианное время выполнения каждого урока среди всех студентов. Потенциальное преимущество медианы состоит в том, что, в отличие от среднего значения, экстремальные значения (выбросы) не влияют на неё. Далее вычисляем среднее время прохождения каждого модуля. Таким образом получаем порог по времени, относительно которого система и будет определять успеваемость студентов: если время меньше или равно порогу, то у студента нет отставания по программе обучения и ставится метка класса «0», в ином случае – «1».

Для каждого студента рассчитывается среднее значение от суммы времени, затраченного на прохождение всех уроков. Студенты могут проходить модули в любой последовательности. Среднее время выполнения пройденных модулей сравнивается со средним пороговым временем выполнения такого же набора модулей.

В конце, для выявления студентов с низкой успеваемостью вычисляется среднее значение меток с округлением значения вниз. Так как предполагается, что анализ активности будет проводиться через определённые промежутки времени, то это позволит определять таких пользователей в регулируемые периоды времени.

При расчёте медианы используется следующая последовательность действий. Первым делом данные времени выполнения ранжируют (сортируют по убыванию). Далее есть два варианта. Если количество значений нечётно, то медиана будет соответствовать центральному значению ряда, номер которого можно определить по формуле:

$$№_{Me} = \frac{N+1}{2}$$

где

$№_{Me}$ – номер значения, соответствующего медиане,

N – количество значений в совокупности данных.

Тогда медиана обозначается, как

$$Me = x_{\frac{N+1}{2}}$$

Это первый вариант, когда в данных есть одно центральное значение. Второй вариант наступает тогда, когда количество данных четно, то есть вместо одного есть два центральных значения. В данном случае берётся средняя арифметическая из двух центральных значений:

$$Me = \frac{x_{\frac{N}{2}} + x_{\frac{N+1}{2}}}{2}$$

4 Практическое исследование решения задачи анализа активностей пользователя

4.1 Исходные данные и их предобработка

В данной работе были использованы следующие средства:

1. Язык программирования Python 3.7.3;
2. Jupyter Notebook (командная оболочка для интерактивных вычислений, в данной работе используется версия на базе языка Python. Оболочка часто используется для работы с данными, статистическим моделированием и машинным обучением);
3. База данных Skillbox (онлайн-университет, откуда была взята информация для обработки и дальнейшего использования);
4. Дополнительные библиотеки:
    - NumPy (библиотека для работы с данными, состоит из вычислительных алгоритмов (в виде функций и операторов), оптимизированных для работы с многомерными массивами);
    - Pandas (библиотека на языке Python для обработки и анализа данных. Работа pandas с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня);
    - Scikit-learn (библиотека машинного обучения);
    - Matplotlib (библиотека для визуализации данных двумерной графикой);
    - Seaborn (библиотека для красивой визуализации данных);
    - Relativedelta (библиотека для расчёта разницы дат);
    - Datetime (библиотека для работы с датами);
    - Calendar (библиотека для перевода числовых значений дат в словесные);
    - Math (библиотека для математических функций, определенных стандартом языка C);
    - Statistics (библиотека для расчета математической статистики числовых данных);
    - CatBoost (библиотека, разработанная компанией Яндекс и реализующая уникальный патентованный алгоритм построения моделей машинного обучения, использующий схему градиентного бустинга).

Данные были скачаны в виде пяти таблиц в формате CSV.

courses.csv содержит следующие значения:

- id – идентификатор курса
- title – название курса
- field – сфера, к которой относится курс

students.csv содержит следующие значения:

- id – идентификатор студента
- city – город студента
- birthday – день рождения студента
- is_working – наличие работы (true/false)
- works_by_profession – работает по профессии (true/false)
- education – степень образования (0 – основное общее; 1 – среднее общее; 2 – среднее профессиональное; 3 – высшее)
- is_married – семейное положение (true/false)
- has_children – наличие детей (true/false)
- work_places_cnt – количество смененных мест работы

course_contents.csv содержит следующие значения:

- course_id – идентификатор курса
- module_number – номер модуля
- module_title – название модуля
- lesson_number – номер урока
- lesson_title – название урока
- lesson_token – токен урока
- is_video – наличие видео (true/false)
- is_homework – наличие домашней работы (true/false)

progresses.csv содержит следующие значения:

- id – идентификатор прогресса
- student_id – идентификатор студента
- course_id – идентификатор курса

progress_phases.csv содержит следующие значения:

- progress_id – идентификатор прогресса

- module_number – номер модуля
- lesson_number – номер урока
- status – статус прохождения урока
- start_date – дата начала
- finish_date – дата окончания

Для удобства дальнейшего использования данные четырёх таблиц (courses.csv, course_contents.csv, progresses.csv, progress_phases.csv) были объединены в ходе выполнения программы в один общий датафрейм **table**.

```
In [4]: table[table.student_id == '768c2987a744c51ce64a5993a2a94eaf'].head()  # проверяем правильность соединения датасетов
```

	progress_id	module_number	lesson_number	status	start_date	finish_date	student_id	
0	a387ab916f402cb3fbfffd29f68fd0ce	2	4	done	2018-06-23 08:28:50.681065+00	2018-06-23 08:28:52.439542+00	768c2987a744c51ce64a5993a2a94eaf	c9fcb746d51e41bc
526	a387ab916f402cb3fbfffd29f68fd0ce	1	1	done	2018-06-20 14:25:21.783762+00	2018-06-20 15:45:07.717209+00	768c2987a744c51ce64a5993a2a94eaf	c9fcb746d51e41bc
2237	a387ab916f402cb3fbfffd29f68fd0ce	2	2	done	2018-06-23 08:18:09.653771+00	2018-06-23 08:18:12.784616+00	768c2987a744c51ce64a5993a2a94eaf	c9fcb746d51e41bc
2788	a387ab916f402cb3fbfffd29f68fd0ce	1	4	done	2018-06-20 16:00:06.36178+00	2018-06-21 19:09:30.845034+00	768c2987a744c51ce64a5993a2a94eaf	c9fcb746d51e41bc

Рис. 4.1.1. Объединённый датафрейм table

Высчитываем основные метрики:

```
courses_count = len(table['course_id'].unique())
print('Общее количество курсов: '+str(courses_count))
```

```
Общее количество курсов: 15
```

Рис. 4.1.2. Число курсов

```python
# группируем таблицу соответствия курсов списку модулей
df_modules = table[['course_title',\
                    'module_title']]\
             .groupby(['course_title'])['module_title']\
             .unique()\
             .to_frame()
# создаём столбец для подсчёта количества модулей в курсе
df_modules['modules_count'] = 0

# проходим по таблице и записываем размерности списков модулей
for i in range (0,df_modules['module_title'].size):
    df_modules['modules_count'][i] = df_modules['module_title'][i].size

print('Количество модулей на каждом курсе:')
print(df_modules['modules_count'])
```

```
Количество модулей на каждом курсе:
course_title
Excel Базовый                            9
Java-разработчик                        17
Java-разработчик с нуля                  9
JavaScript с нуля                       18
PHP-разработчик с 0 до PRO. Часть 1      8
SMM-маркетолог от А до Я                11
UX-дизайн                               20
Анимация интерфейсов                    21
Веб-вёрстка для начинающих 2.0           8
Веб-дизайн PRO 2.0                      17
Веб-дизайн Базовый                      17
Веб-дизайн с нуля 2.0                   19
Веб-разработчик                         20
Интернет-маркетолог от Ingate           18
Руководитель digital-проектов           17
Name: modules_count, dtype: int64
```

Рис. 4.1.3. Подсчёт модулей в каждом курсе

```python
 # группируем таблицу соответствия курсов и модулей списку уроков
df_lessons = table[['course_title',\
                    'module_title',\
                    'lesson_title']]\
                   .groupby(['course_title', 'module_title'])\
                   ['lesson_title']\
                   .unique()\
                   .to_frame()
# создаём столбец для подсчёта количества уроков в модуле
df_lessons['lessons_count'] = 0

# проходим по таблице и записываем размерности списков уроков
for i in range (0,df_lessons['lesson_title'].size):
    df_lessons['lessons_count'][i] = df_lessons['lesson_title'][i].size

df_lessons.drop(columns='lesson_title', inplace=True)

print('Количество уроков в каждом модуле на каждом курсе:')
print(df_lessons)
```

```
Количество уроков в каждом модуле на каждом курсе:
                                                                lessons_count
course_title                module_title
Excel Базовый                Визуализация данных Excel                      5
                            Основной функционал Excel                      11
                            Основной функционал Excel (продолжение)         7
                            Сводные таблицы Excel                           5
                            Формулы и функции Excel. Более сложные формулы  5
...                                                                       ...
Руководитель digital-проектов Решение факапов. Lean/TOC. Обзор.             5
                            Требовательность digital-продюсера              4
                            Управление временем                             4
                            Управление дизайнерами. Разработка дизайна по s...  7
                            Экологичный путь менеджера                      4
```

Рис. 4.1.4. Подсчёт уроков в каждом модуле всех курсов

```python
df_median = df_lessons.groupby(['course_title'])['lessons_count'].median()
print('Медианное количество уроков в модуле на каждом курсе:')
print(df_median)
```

```
Медианное количество уроков в модуле на каждом курсе:
course_title
Excel Базовый                        5.0
Java-разработчик                     7.0
Java-разработчик с нуля             10.0
JavaScript с нуля                    7.0
PHP-разработчик с 0 до PRO. Часть 1  4.0
SMM-маркетолог от А до Я             6.0
UX-дизайн                            3.5
Анимация интерфейсов                 3.0
Веб-вёрстка для начинающих 2.0       7.0
Веб-дизайн PRO 2.0                   5.0
Веб-дизайн Базовый                   3.0
Веб-дизайн с нуля 2.0                4.0
Веб-разработчик                      2.0
Интернет-маркетолог от Ingate        6.5
Руководитель digital-проектов        5.0
```

Рис. 4.1.5. Медиана суммарного количества уроков в модулях на курсах

```
 # группируем таблицу соответствия курсов списку идентификаторов студентов
df_students = table[['course_title',\
                     'student_id']]\
                   .groupby(['course_title'])\
                    ['student_id'].\
                    unique().\
                    to_frame()
# создаём столбец для подсчёта количества студентов на курсе
df_students['students_count'] = 0

# проходим по таблице и записываем размерности списков студентов
for i in range (0,df_students['student_id'].size):
    df_students['students_count'][i] = df_students['student_id'][i].size
# обновляем индекс
df_students.reset_index(drop=False, inplace=True)

print('Количество учеников на каждом курсе:')
print(df_students[['course_title', 'students_count']])
```

```
Количество учеников на каждом курсе:
                          course_title  students_count
0                        Excel Базовый             782
1                     Java-разработчик             763
2                Java-разработчик с нуля             581
3                     JavaScript с нуля             966
4   PHP-разработчик с 0 до PRO. Часть 1             854
5              SMM-маркетолог от А до Я             506
6                             UX-дизайн            1151
7                  Анимация интерфейсов             598
8        Веб-вёрстка для начинающих 2.0            2004
9                    Веб-дизайн PRO 2.0            1711
10                   Веб-дизайн Базовый             518
11                Веб-дизайн с нуля 2.0            2014
12                     Веб-разработчик             628
13       Интернет-маркетолог от Ingate            2168
14       Руководитель digital-проектов             685
```

Рис. 4.1.6. Число студентов по курсам

```
# вычисляем сегодняшнюю дату
today = datetime.date.today()
# вычисляем возраст каждого студента
students['age'] = [get_age(i, today) for i in students['birthday']]
```

```
print('Минимальный возраст: '+str(students['age'].min())+' лет')
print('Максимальный возраст: '+str(students['age'].max())+' лет')
print('Средний возраст: '+str(students['age'].mean())+' лет')
print('Медианный возраст: '+str(students['age'].median())+' лет')
```

```
Минимальный возраст: 18 лет
Максимальный возраст: 67 лет
Средний возраст: 34.34861966689482 лет
Медианный возраст: 34.0 лет
```

Рис. 4.1.7. Статистика возрастов

```
# создаём столбец в датафрейме df_students для списков возрастов студентов каждого курса
df_students['ages'] = np.empty((len(df_students), 0)).tolist()

# проходим по таблице студентов
for i in range(len(df_students['student_id'])):
    for j in range(len(df_students['student_id'][i])):
        # добавляем возраст каждого студента в список возрастов студентов на курсе
        df_students['ages'][i].append(students[students['id']==df_students['student_id'][i][j]]['age'].values[0])
```

```
# проходим удаляем из столбца возрастов отсутствующие значения
for i in range(len(df_students)):
    df_students['ages'][i] = [x for x in df_students['ages'][i] if str(x) != 'nan']
```

```
# создаём столбец в датафрейме df_students для минимального возраста студентов каждого курса
df_students['min_age'] = float('NaN')

# создаём столбец в датафрейме df_students для максимального возраста студентов каждого курса
df_students['max_age'] = float('NaN')

# создаём столбец в датафрейме df_students для среднего возраста студентов каждого курса
df_students['mean_age'] = float('NaN')

# создаём столбец в датафрейме df_students для медианного возраста студентов каждого курса
df_students['median_age'] = float('NaN')

# вычисляем минимальный, максимальный, средний, медианный возраст студентов на каждом курсе
for i in range(len(df_students)):
    df_students['min_age'][i] = min(df_students['ages'][i])
    df_students['max_age'][i] = max(df_students['ages'][i])
    df_students['mean_age'][i] = np.mean(df_students['ages'][i])
    df_students['median_age'][i] = np.median(df_students['ages'][i])
```

Рис. 4.1.8. Подсчёт статистики возрастов студентов на каждом курсе

```
df_students.drop(columns=['students_count','student_id','ages'])
```

	course_title	min_age	max_age	mean_age	median_age
0	Excel Базовый	21.0	48.0	25.406650	25.0
1	Java-разработчик	27.0	52.0	39.702490	39.0
2	Java-разработчик с нуля	30.0	62.0	45.452668	45.0
3	JavaScript с нуля	24.0	48.0	35.670807	36.0
4	PHP-разработчик с 0 до PRO. Часть 1	21.0	56.0	36.322014	36.0
5	SMM-маркетолог от А до Я	24.0	54.0	35.432806	35.0
6	UX-дизайн	22.0	45.0	29.881842	30.0
7	Анимация интерфейсов	22.0	47.0	29.732441	29.0
8	Веб-вёрстка для начинающих 2.0	22.0	49.0	34.021956	34.0
9	Веб-дизайн PRO 2.0	18.0	51.0	32.680888	32.0
10	Веб-дизайн Базовый	24.0	62.0	35.926641	35.0
11	Веб-дизайн с нуля 2.0	25.0	50.0	34.795929	35.0
12	Веб-разработчик	23.0	48.0	34.119427	34.0
13	Интернет-маркетолог от Ingate	23.0	67.0	44.175738	44.0
14	Руководитель digital-проектов	27.0	50.0	34.575182	34.0

Рис. 4.1.9. Статистика возрастов студентов на каждом курсе

Построим столбчатую диаграмму и отобразим на ней суммарное количество пользователей на разных направлениях подготовки. Значения отсортированы по

возрастанию. Цвет столбцов содержит информацию о сфере, к которой относится курс. На график также нанесена линия медианы красного цвета.

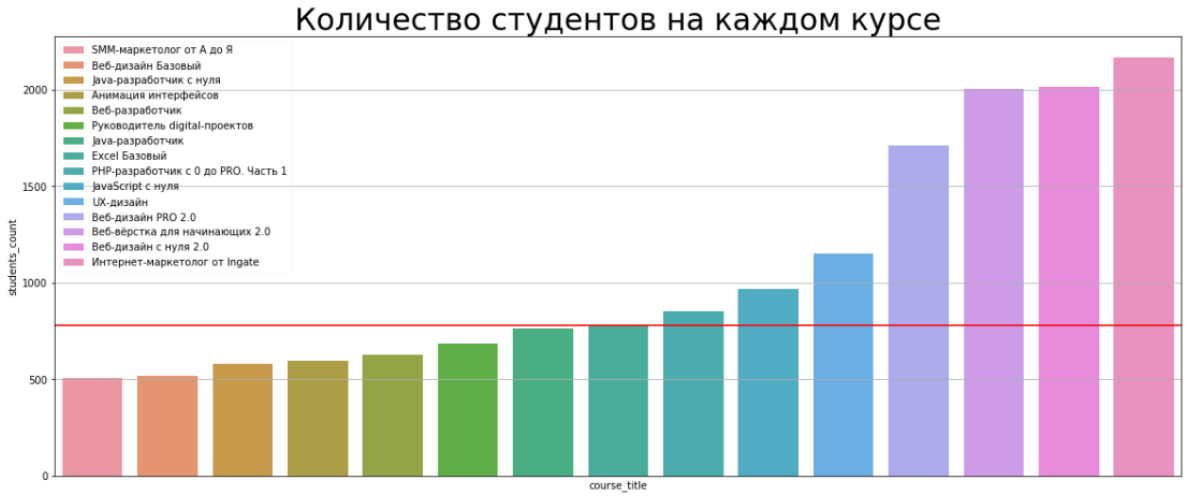

Рис. 4.1.10. График количества студентов на каждом курсе

На основании рассчитанных значений имеем следующие данные и метрики.

courses_count – количество уникальных идентификаторов курсов, оно же и есть количество курсов

df_modules содержит следующие значения:

course_title – название курса в качестве идентификатора

module_title – массив названий всех модулей курса

modules_count – количество модулей в курсе

df_lessons содержит следующие значения:

course_title – название курса в качестве 1-ого идентификатора

module_title – название модуля в качестве 2-ого идентификатора

lessons_count – количество уроков в модуле

df_median содержит следующие значения:

course_title – название курса в качестве идентификатора

lessons_count – медианное количество уроков в модуле на каждом курсе

В таблицу **students** были добавлены следующие значения:

age – вычисленный возраст студента типа float

df_students содержит следующие значения:

course_title – название курса

student_id – массив идентификаторов студентов курса

students_count – количество студентов на курсе

ages – массив возрастов студентов курса

min_age – минимальный возраст среди студентов курса

max_age – максимальный возраст среди студентов курса

mean_age – средний возраст среди студентов курса

median_age – медианный возраст среди студентов курса

median_students_count – медиана количества студентов

4.2 Описание экспериментальных исследований

Расчёт потенциальной нагрузки на преподавателей

Для того чтобы понять, насколько активны пользователи системы, рассчитаем рабочую нагрузку на преподавателей:

Рассчитываем прирост студентов на каждом курсе в каждом месяце за всю историю (каждый месяц в диапазоне от марта 2016 до июля 2019 включительно). Будем считать дату начала прохождения курса студентом по дате начала первой домашней работы.

Рассчитываем количество новых пользователей на учебном портале в течение всего времени существования системы. Будем брать дату начала выполнения 1-ого д/з в качестве точки отсчёта прохождения курса.

```
# создаём датафрэйм, в котором собрали id всех студентов по каждому курсу
# и в который будем производить дальнейшие записи
df_growth = table[['course_title', 'course_id', 'student_id']]\
            .groupby(['course_title', 'course_id'])['student_id']\
            .unique()\
            .to_frame()\
            .reset_index()
```

Рис. 4.2.1. Формирование датафрейма df_growth

```
# создаём пустой список, в который будем записывать списки
# с датами начала первой домашней работы по каждому курсу
dates_list = []
for i in df_growth['course_id']:
    print(i)
    for j in df_growth[df_growth['course_id']==i]['student_id']:
        print(j)
        buf_list = []
        for k in j:
            try:
                buf_list.append(np.nanmin(table[(table['course_id']==i)&\
                                                (table['is_homework']==True)&\
                                                (table['student_id']==k)]['start_date']))
            except ValueError:
                continue
        dates_list.append(buf_list)
```

Рис. 4.2.2. Создание списка дат первого д/з

```
# записываем все даты в датафрэйм
df_growth['dates'] = dates_list

# выделяем из дат год и месяц и записываем всё в датафрэйм
df_growth['months'] = ''
for i in range(len(df_growth)):
    df_growth['months'][i]=[x[0:7] for x in df_growth.iloc[i]['dates']]

# создаём список из месяцов в диапазоне от марта 2016 до июля 2019 включительно
list_of_dates = [x[0:7] for x in np.datetime_as_string(pd.date_range('2016-03', '2019-08', freq='M').values)]

# высчитываем прирост студентов на каждом курсе в каждом месяце
for i in list_of_dates:
    list_of_counts = []
    for j in df_growth.index:
        list_of_counts.append(df_growth['months'][j].count(i))
    df_growth[i] = list_of_counts
```

Рис. 4.2.3. Вычисление прироста студентов

```
df_growth
```

student_id	dates	months	2016-03	2016-04	2016-05	2016-06	2016-07	...	2018-10	2018-11	2018-12	2019-01	2019-02	2019-03	2019-04	2019-05
[2364226ef29a6afef154dc8f5cef203d, c8620d7c866...	[2018-12-10 16:15:15.891656+00, 2019-05-01 18:...	[2018-12, 2019-05, 2018-11, 2019-04, 2018-08, ...	0	0	0	0	0	...	30	33	43	52	46	63	46	45
[fb019632313c642191d5a072499ad647, 525cd8878b2...	[2018-07-16 18:24:26.22924+00, 2018-05-06 10:4...	[2018-07, 2018-05, 2018-11, 2019-01, 2019-01, ...	0	0	0	0	0	...	33	32	102	75	50	74	7	4
[9f5b2a1edcb6a434bf165fb26053c346, 6303921b52b...	[2019-07-15 13:56:59.859533+00, 2019-06-17 17:...	[2019-07, 2019-06, 2019-04, 2019-06, 2019-	0	0	0	0	0	...	0	0	0	0	0	74	109	74

Рис. 4.2.4. Датафрейм прироста количества студентов по месяцам

На основании предыдущего расчёта строим линейную диаграмму, где каждая линия показывает количество новых студентов на курсе в течение всей истории работы сервиса. Линия для каждого курса имеет свой цвет, что отображено в легенде.

Рис. 4.2.5. График прироста студентов на каждом курсе

При рассмотрении прироста количества студентов на курсах по месяцам можно заметить на графике (*Рис. 4.2.5)*, что до января 2018 г. прироста практически не было. Достаточно сложно сказать почему это происходило, ведь не известно добавлялись все курсы сразу или постепенно, поэтому предположим, что все курсы были сразу доступны. А значит, наличие такого периода, скорее всего, связано с малой известностью сервиса на тот момент. Однако после этого отмечается равномерный наплыв пользователей, вероятно, связанный с началом более активной рекламой кампанией онлайн-университета в сети. Пик приходится на период с декабря 2018 по январь 2019. Скорее всего, в то время сервис устроил акцию в честь праздников и предоставлял выгодные условия на приобретение курсов. Самый популярный курс – это «Веб-дизайн с нуля 2.0»; самый непопулярный – «Анимация интерфейсов». Между курсами «Веб-дизайн Базовый» и «Веб-дизайн с нуля 2.0» наблюдается обратно-пропорциональная зависимость количества новых студентов, что происходит, если судить по названиям, из-за ненадобности первого, ведь он должен быть включён в состав второго. Аналогичная ситуация происходит с курсами «Java-разработчик» и «Java-разработчик с нуля». Если проделанная

аналитика верна, то увеличение количества человек на платформе напрямую связано с маркетинговыми ходами. А вот популярность устаревших версий курсов резко начинает снижаться после выхода обновлённой версии.

Теперь рассчитаем количество прогрессов по выполнению домашних работ в каждом месяце за всю историю (каждый месяц в диапазоне от марта 2016 до июля 2019 включительно) для каждого курса. Здесь следует учитывать, что выполнение домашнего задания может перетекать из одного месяца в другой (такие дз включены в общее число прогрессов для всех месяцев, которые покрывает срок выполнения этих дз)

```
# создаём датафрэйм, в котором собрали id всех прогрессов и даты их выполнения по каждому курсу
df_progress = table[(table['is_homework']==True)&\
               (table['status']=='done')][['course_title',\
                                           'progress_id',\
                                           'start_date',\
                                           'finish_date']]
df_progress
```

	course_title	progress_id	start_date	finish_date
3555	Веб-дизайн PRO 2.0	a387ab916f402cb3fbfffd29f68fd0ce	2018-06-21 19:10:36.957891+00	2018-06-28 15:59:25.320418+00
3556	Веб-дизайн PRO 2.0	3b9dce04f32da32763124602557f92a3	2019-03-16 15:28:29.978311+00	2019-03-18 09:51:16.562395+00
3557	Веб-дизайн PRO 2.0	73e17a05355852fe65b785c82c37d1ad	2019-07-02 19:01:15.282595+00	2019-07-03 18:10:52.187797+00
3558	Веб-дизайн PRO 2.0	cc3eb34ae49c719648352c4175daee88	2018-07-25 09:39:56.674653+00	2018-07-27 15:39:17.61395+00
3559	Веб-дизайн PRO 2.0	04ace4fe130d90c801e24eea13ee808e	2019-04-24 18:11:57.82383+00	2019-04-25 10:11:25.756062+00
...	...	...	...	...
350655	Java-разработчик с нуля	45289b36f4827d49d9d3757fc9486ca0	2019-06-30 12:24:44.841887+00	2019-07-01 05:11:00.227931+00
350656	Java-разработчик с нуля	7c46c781a54bf9d17f24b4920df4f7eb	2019-06-29 11:22:10.230328+00	2019-07-02 20:09:44.150192+00
350657	Java-разработчик с нуля	56a9cb904c20cd2fcfa8104f87f988a7	2019-07-03 05:03:29.121944+00	2019-07-04 07:53:36.550522+00
350670	Java-разработчик с нуля	f6216204f2c65eea37f913cfa5e4eb6a	2019-07-01 20:06:56.697121+00	2019-07-06 04:47:43.213367+00
350671	Java-разработчик с нуля	45289b36f4827d49d9d3757fc9486ca0	2019-07-01 14:35:47.629736+00	2019-07-12 03:36:51.775851+00

44926 rows × 4 columns

Рис. 4.2.6. Формирование датафрейма df_progress

```
# высчитываем количество прогрессов в каждом курсе
df_progress_task_1 = df_progress.drop(columns=['progress_id',\
                                'start_date',\
                                'finish_date',\
                                'progress_months']).groupby('course_title')[list_of_dates].sum().reset_index()
df_progress_task_1
```

	course_title	2016-03	2016-04	2016-05	2016-06	2016-07	2016-08	2016-09	2016-10	2016-11	...	2018-10	2018-11	2018-12	2019-01	2019-02	2019-03	2019-04	2019-05	2019-06	2019-07
0	Excel Базовый	0	0	0	0	0	0	0	0	0	...	235	224	227	311	327	340	305	289	246	157
1	Java-разработчик	0	0	0	0	0	0	0	0	0	...	240	226	377	449	427	469	261	171	102	36
2	Java-разработчик с нуля	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	87	477	637	924	616
3	JavaScript с нуля	0	0	0	0	0	0	0	0	0	...	158	175	201	318	315	327	315	308	296	101
4	PHP-разработчик с 0 до PRO. Часть 1	0	0	0	0	0	0	0	0	0	...	120	97	135	142	97	122	144	130	82	29
5	SMM-маркетолог от А до Я	0	0	0	0	0	0	0	0	0	...	32	57	62	84	133	134	160	159	182	127
6	UX-дизайн	0	0	0	0	0	0	0	0	4	...	303	298	297	351	328	303	228	195	200	86
7	Анимация интерфейсов	0	0	0	0	0	0	0	0	0	...	168	159	127	185	158	141	92	80	72	25
8	Веб-вёрстка для начинающих 2.0	0	0	0	0	0	0	0	0	0	...	396	431	507	569	502	532	473	408	348	141
9	Веб-дизайн PRO 2.0	0	0	0	0	0	0	0	0	0	...	280	277	290	335	380	487	448	477	452	173
10	Веб-дизайн Базовый	0	0	0	0	0	0	0	0	0	...	248	212	178	186	143	109	98	66	45	24
11	Веб-дизайн с нуля 2.0	0	0	0	0	0	0	0	0	0	...	938	1140	1371	2130	1975	1919	1716	1394	1241	458
12	Веб-разработчик	5	0	0	0	0	0	0	0	0	...	138	144	96	96	104	74	57	51	41	16
13	Интернет-маркетолог от Ingate	0	0	0	0	0	0	0	0	0	...	592	678	691	820	770	751	684	609	618	296
14	Руководитель digital-проектов	0	0	0	0	0	0	0	0	0	...	301	268	270	347	348	348	264	256	172	103

15 rows × 42 columns

Рис. 4.2.7. Датафрейм прогресса в обучении по месяцам

Строим один line-graph для всех курсов по последнему расчёту (Рис. 4.2.8). 15 линий на графике. Линия для каждого курса имеет свой цвет, что отображено в легенде.

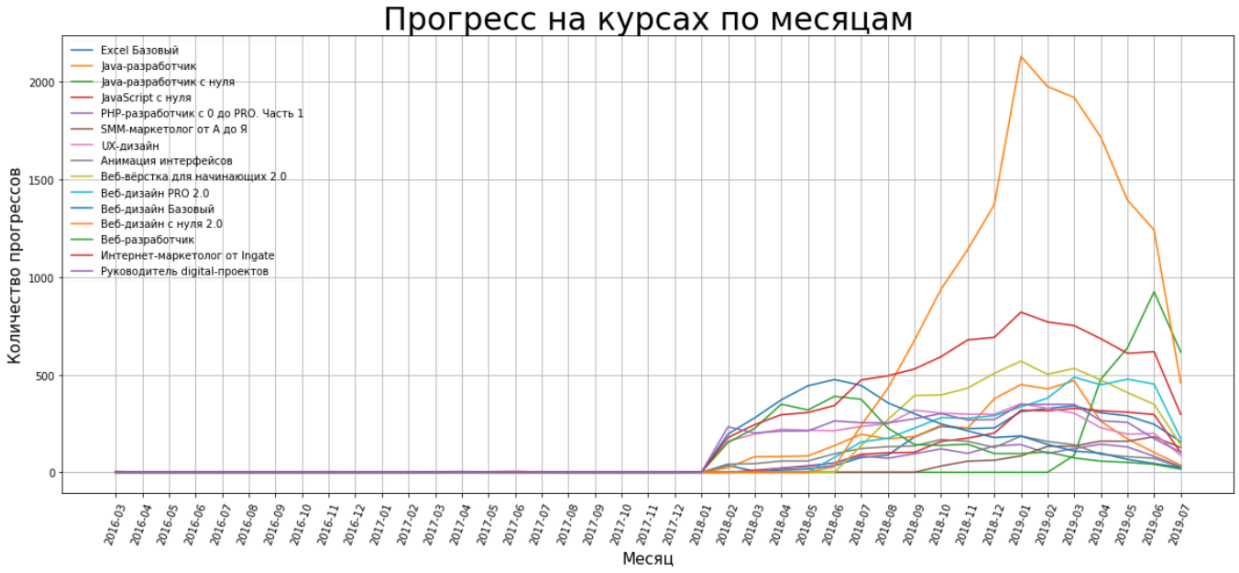

Рис. 4.2.8. График прогресса студентов на каждом курсе

При рассмотрении прогресса можно заметить на графике, что до января 2018 г. прогрессов по выполнению домашних заданий не было, а соответственно и нагрузки на проверяющих тоже. С увеличением количества пользователей нагрузка на преподавателей начала постепенно увеличиваться. С января по март 2019 г. наблюдаем большое количество прогрессов, что связано как с возросшим числом студентов, так и с освободившимся временем у студентов (вероятно, отпуск или каникулы).

Выявление проблемных модулей

Чтобы выявить сезонность, посчитаем медианное время выполнения домашней работы по месяцам (12 месяцев, январь-декабрь) для каждого курса.

```
# создаём датафрейм, в котором собрали время выполнения всех д/з
df_time = table[(table['is_homework']==True)&\
              (table['status']=='done')][['course_title',\
                                          'module_number',\
                                          'student_id',\
                                          'start_date',\
                                          'finish_date']]\
                                                      .reset_index(drop=True)
```

Рис. 4.2.9. Формирование датафрейма df_time

```
# преобразуем строковый тип даты к типу  datetime.datetime
df_time['start_date'] = [datetime.datetime.strptime(df_time['start_date'][i][0:18],\
                                                    "%Y-%m-%d %H:%M:%S") for i in df_time.index]

# преобразуем строковый тип даты к типу  datetime.datetime
df_time['finish_date'] = [datetime.datetime.strptime(df_time['finish_date'][i][0:18],\
                                                     "%Y-%m-%d %H:%M:%S") for i in df_time.index]

# вычисляем разность между временем начала и окончания выполнения домашней работы
df_time['spent_time'] = [relativedelta(df_time['finish_date'][i], df_time['start_date'][i]) for i in df_time.index]

# переводим время в секунды
df_time['spent_sec'] = [(df_time['spent_time'][x].days*24*60*60\
                         +df_time['spent_time'][x].hours*60*60\
                         +df_time['spent_time'][x].minutes*60\
                         +df_time['spent_time'][x].seconds) for x in range(len(df_time))]
```

Рис. 4.2.10. Вычисление времени выполнения д/з в секундах

```
df_time
```

	course_title	module_number	student_id	start_date	finish_date	spent_time	spent_sec
0	Веб-дизайн PRO 2.0	1	768c2987a744c51ce64a5993a2a94eaf	2018-06-21 19:10:03	2018-06-28 15:59:02	relativedelta(days=+6, hours=+20, minutes=+48,...	593339
1	Веб-дизайн PRO 2.0	1	03151bc73bdb29fe1be1443c6d83e22f	2019-03-16 15:28:02	2019-03-18 09:51:01	relativedelta(days=+1, hours=+18, minutes=+22,...	152579
2	Веб-дизайн PRO 2.0	1	ed235f47e16da6e83d3f1cb511f38ea6	2019-07-02 19:01:01	2019-07-03 18:10:05	relativedelta(hours=+23, minutes=+9, seconds=+4)	83344
3	Веб-дизайн PRO 2.0	1	59e8681cb7b5c8043ae1aac10c8053ca	2018-07-25 09:39:05	2018-07-27 15:39:01	relativedelta(days=+2, hours=+5, minutes=+59, ...	194396
4	Веб-дизайн PRO 2.0	1	c16250079190337fe9074736e33eecb2	2019-04-24 18:11:05	2019-04-25 10:11:02	relativedelta(hours=+15, minutes=+59, seconds=...	57597
...	...	...	...	...	...	...	...
44921	Java-разработчик с нуля	9	4e5f1ba884ba5759c07cf6e942ae5e98	2019-06-30 12:24:04	2019-07-01 05:11:00	relativedelta(hours=+16, minutes=+46, seconds=...	60416
44922	Java-разработчик с нуля	9	1a0acf593d0a38155bafdf29e3cba338	2019-06-29 11:22:01	2019-07-02 20:09:04	relativedelta(days=+3, hours=+8, minutes=+47, ...	290823
44923	Java-разработчик с нуля	9	ed93dfa830d97cf67eb047dd7a4aa181	2019-07-03 05:03:02	2019-07-04 07:53:03	relativedelta(days=+1, hours=+2, minutes=+50, ...	96601
44924	Java-разработчик с нуля	9	c21fa1f7fa61d11253e4cc8eeeb03027	2019-07-01 20:06:05	2019-07-06 04:47:04	relativedelta(days=+4, hours=+8, minutes=+40, ...	376859
44925	Java-разработчик с нуля	9	4e5f1ba884ba5759c07cf6e942ae5e98	2019-07-01 14:35:04	2019-07-12 03:36:05	relativedelta(days=+10, hours=+13, minutes=+1,...	910861

44926 rows × 7 columns

Рис. 4.2.11. Датафрейм с потраченным на выполнение д/з временем

```
df_time_task_3 = df_time[['course_title','module_number','start_date','finish_date','spent_sec']].copy()
df_time_task_3['month'] = pd.DatetimeIndex(df_time_task_3['start_date']).month
df_time_task_3
```

	course_title	module_number	start_date	finish_date	spent_sec	month
0	Веб-дизайн PRO 2.0	1	2018-06-21 19:10:03	2018-06-28 15:59:02	593339	6
1	Веб-дизайн PRO 2.0	1	2019-03-16 15:28:02	2019-03-18 09:51:01	152579	3
2	Веб-дизайн PRO 2.0	1	2019-07-02 19:01:01	2019-07-03 18:10:05	83344	7
3	Веб-дизайн PRO 2.0	1	2018-07-25 09:39:05	2018-07-27 15:39:01	194396	7
4	Веб-дизайн PRO 2.0	1	2019-04-24 18:11:05	2019-04-25 10:11:02	57597	4
...	...	...	...	...	...	...
44921	Java-разработчик с нуля	9	2019-06-30 12:24:04	2019-07-01 05:11:00	60416	6
44922	Java-разработчик с нуля	9	2019-06-29 11:22:01	2019-07-02 20:09:04	290823	6
44923	Java-разработчик с нуля	9	2019-07-03 05:03:02	2019-07-04 07:53:03	96601	7
44924	Java-разработчик с нуля	9	2019-07-01 20:06:05	2019-07-06 04:47:04	376859	7
44925	Java-разработчик с нуля	9	2019-07-01 14:35:04	2019-07-12 03:36:05	910861	7

44926 rows × 6 columns

Рис. 4.2.12. Формирование датафрейма df_time_task_3 с номером месяца начала выполнения д/з

```
df_for_plot_2 = pd.DataFrame(df_time_task_3.groupby(['course_title', 'month'])['spent_sec'].median())
df_for_plot_2.rename(columns = {'spent_sec':'median'}, inplace = True)
df_for_plot_2 = df_for_plot_2.reset_index(['course_title', 'month'])\
                            .set_index('course_title')[['month','median']]
df_for_plot_2
```

course_title	month	median
Excel Базовый	1	188220.0
Excel Базовый	2	192780.0
Excel Базовый	3	244083.0
Excel Базовый	4	167611.5
Excel Базовый	5	208528.5
...	...	...
Руководитель digital-проектов	8	820499.0
Руководитель digital-проектов	9	763685.0
Руководитель digital-проектов	10	404398.0
Руководитель digital-проектов	11	502410.5
Руководитель digital-проектов	12	645540.0

171 rows × 2 columns

Рис. 4.2.12. Формирование датафрейма df_for_plot_2 с номером месяца начала выполнения д/з и медианным временем выполнения

На основании датафрейма df_for_plot_2 построим line-graph, на который будут нанесены линии для каждого курса с медианным временем выполнения домашней работы по месяцам. 15 линий на графике. Линия для каждого курса имеет свой цвет,

что отображено в легенде.

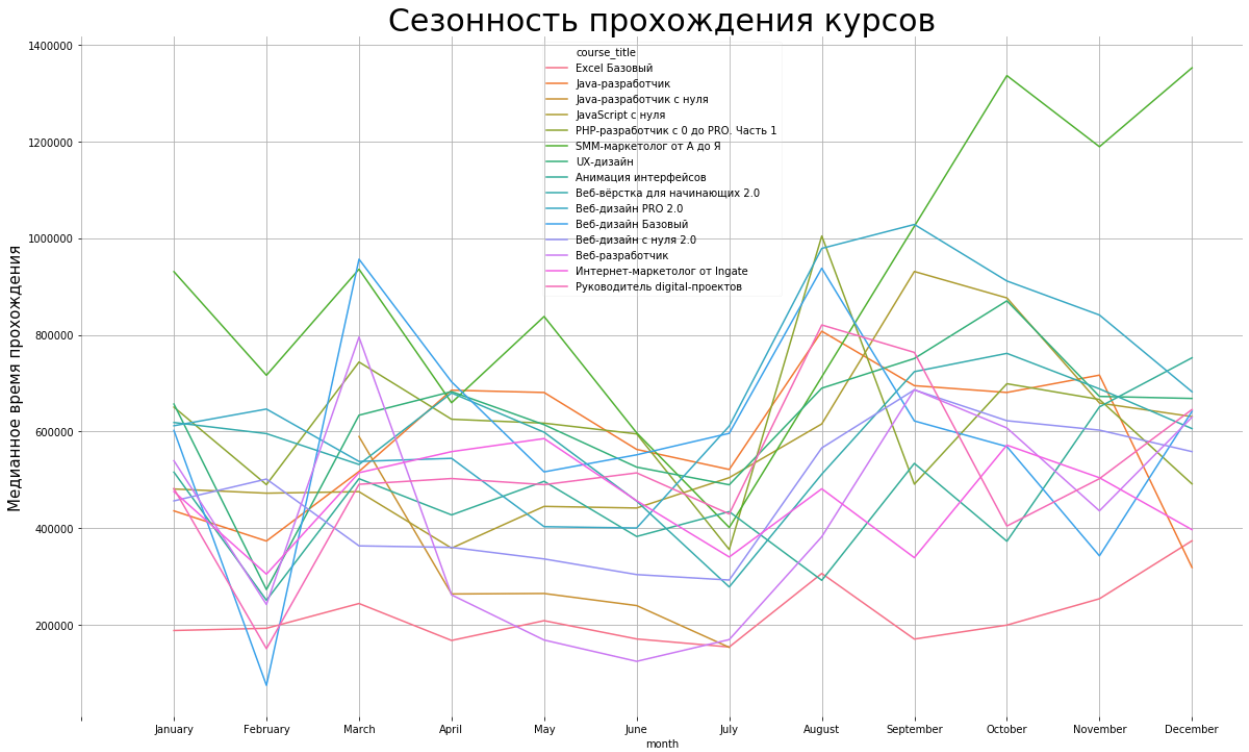

Рис. 4.2.13. График сезонности прохождения курсов

При рассмотрении графика сезонности прохождения курсов в течение года (*Рис. 4.2.13)* можно заметить, что в феврале наблюдается уменьшение медианного временем выполнения домашнего задания по всем направлениям подготовки, а в марте наоборот – увеличение. Аналогичную ситуацию можно увидеть в июле и августе: в июле медианное время уменьшается по всем курсам, в августе – резко возрастает. Если вспомнить выводы, сделанные при анализе графика прогресса студентов на каждом курсе, то можно прийти к выводу, что сезонность прохождения курсов напрямую связана с сезонами каникул и отпусков, когда у пользователей появляется больше свободного времени на выполнение заданий. Если же рассматривать каждый курс по отдельности, то из общей массы выделяются «Excel Базовый», как направление подготовки с самым маленьким медианным временем, и «SMM-маркетолог от А до Я», как направление, на которое студенты тратят больше всего времени. В последнем случае стоит обратить внимание на сложность заданий, с которыми студенты очевидно справляются не так успешно.

Метрика успеваемости

Иногда студенты берут курсы, которые оказываются для них неподъемными. Это может быть как по причинам недостаточной изначальной подготовки, так и по причинам, связанным с низкой мотивацией студента. Для улучшения качества контента полезно выявить причину. Ранее, при разработке методов анализа активностей пользователя, уже описывалась метрика успеваемости. На основании метрики успеваемости выявим таких студентов на каждом курсе, чтобы предоставить продюсерам список пользователей, которые отстают по программе обучения.

```
# создаём датафрейм, в котором собрали время выполнения всех заданий
df_metric = table[(table['status']=='done')][['course_title',\
                                              'module_number',\
                                                 'lesson_number',\
                                                 'student_id',\
                                              'start_date',\
                                              'finish_date']]\
                                                          .reset_index(drop=True)
```

Рис. 4.2.14. Формирование датафрейма df_metrics

```
# преобразуем строковый тип даты к типу  datetime.datetime
df_metric['start_date'] = [datetime.datetime.strptime(df_metric['start_date'][i][0:18],\
                                                      "%Y-%m-%d %H:%M:%S") for i in df_metric.index]

# преобразуем строковый тип даты к типу  datetime.datetime
df_metric['finish_date'] = [datetime.datetime.strptime(df_metric['finish_date'][i][0:18],\
                                                       "%Y-%m-%d %H:%M:%S") for i in df_metric.index]

# вычисляем разность между временем начала и окончания прохождения урока
df_metric['spent_time'] = [relativedelta(df_metric['finish_date'][i],\
                                         df_metric['start_date'][i]) for i in df_metric.index]
# переводим затраченное время в секунды
df_metric['spent_sec'] = [(df_metric['spent_time'][x].days*24*60*60\
                           +df_metric['spent_time'][x].hours*60*60\
                           +df_metric['spent_time'][x].minutes*60\
                           +df_metric['spent_time'][x].seconds) for x in range(len(df_metric))]
```

Рис. 4.2.15. Вычисление затраченного времени прохождения в секундах

```
df_metric.head()
```

	course_title	module_number	lesson_number	student_id	start_date	finish_date	spent_time	spent_sec
0	Веб-дизайн PRO 2.0	2	4	768c2987a744c51ce64a5993a2a94eaf	2018-06-23 08:28:05	2018-06-23 08:28:05	relativedelta()	0
1	Веб-дизайн PRO 2.0	2	4	03151bc73bdb29fe1be1443c6d83e22f	2019-03-18 14:23:01	2019-03-18 14:54:05	relativedelta(minutes=+31, seconds=+4)	1864
2	Веб-дизайн PRO 2.0	2	4	ed235f47e16da6e83d3f1cb511f38ea6	2019-07-09 09:18:04	2019-07-11 08:03:00	relativedelta(days=+1, hours=+22, minutes=+44,...	168296
3	Веб-дизайн PRO 2.0	2	4	59e8681cb7b5c8043ae1aac10c8053ca	2018-07-27 15:39:01	2018-07-27 16:13:05	relativedelta(minutes=+34, seconds=+4)	2044
4	Веб-дизайн PRO 2.0	2	4	c16250079190337fe9074736e33eecb2	2019-04-24 18:42:04	2019-04-24 18:44:05	relativedelta(minutes=+2, seconds=+1)	121

Рис. 4.2.16. Датафрейм с потраченным на прохождение уроков временем

```
# вычисляем среднее время, которое конкретный студент тратит на выполнение модуля
df_metric_task_1 = pd.DataFrame(df_metric.groupby(['course_title', 'module_number','student_id'])['spent_sec'].mean())
df_metric_task_1.rename(columns={'spent_sec':'mean'}, inplace=True)
df_metric_task_1.reset_index(['course_title', 'module_number','student_id'], inplace=True)
```

Рис. 4.2.17. Формирование датафрейма df_metrics_task_1

```
# вычисляем медианное время, которое студенты тратят на выполнение каждого урока
df_metric_task_2 = pd.DataFrame(df_metric.groupby(['course_title', 'module_number', 'lesson_number'])['spent_sec'].median())
df_metric_task_2.rename(columns={'spent_sec':'median'}, inplace=True)
df_metric_task_2.reset_index(['course_title', 'module_number', 'lesson_number'], inplace=True)
```

Рис. 4.2.18. Формирование датафрейма df_metrics_task_2

```
# вычисляем среднее время, которое студенты тратят на выполнение каждого модуля
df_metric_task_3 = pd.DataFrame(df_metric_task_2.groupby(['course_title', 'module_number'])['median'].mean())
df_metric_task_3.rename(columns={'median':'mean_median'}, inplace=True)
df_metric_task_3.reset_index(['course_title', 'module_number'], inplace=True)
```

Рис. 4.2.19. Формирование датафрейма df_metrics_task_3

```
df_metric_task_3
```

	course_title	module_number	mean_median
0	Excel Базовый	1	24517.000000
1	Excel Базовый	2	23743.785714
2	Excel Базовый	3	54148.833333
3	Excel Базовый	4	43727.071429
4	Excel Базовый	5	51496.000000
...	...	...	...
224	Руководитель digital-проектов	13	207870.250000
225	Руководитель digital-проектов	14	90706.357143
226	Руководитель digital-проектов	15	18186.958333
227	Руководитель digital-проектов	16	73893.875000
228	Руководитель digital-проектов	17	4200.250000

229 rows × 3 columns

Рис. 4.2.20. Датафрейм со средне-медианным временем выполнения модулей

```
# создаём список студентов, которые приобрели хотя бы один курс
stud_paid_list = list(df_metric['student_id'].unique())
# формируем датафрейм для статусов студентов
status_df = pd.DataFrame(stud_paid_list, columns=['id'])
# записываем курсы, которые у них есть
status_df['courses'] = [[y for y in df_metric_task_1[df_metric_task_1['student_id']==x]['course_title'].unique()] for x in stud_p
```

Рис. 4.2.21. Формирование датафрейма status_df

```
total_list = []
list_len = len(stud_paid_list)
# проходим посписку студентов с оплаченными курсами
for i in stud_paid_list:
    # проходим по всем курсам, что есть у этого студента
    for j in status_df[status_df['id']==i]['courses']:
        curr_list = []
        # находим все модули, которые студент уже выполнил
        for k in j:
            modules_num = list(df_metric_task_1[(df_metric_task_1['student_id']==i)\
                                                &(df_metric_task_1['course_title']==k)]['module_number'].values)
            summary = 0
            # суммируем средние медианные значения времени по пройденным модулям
            for l in modules_num:
                summary += df_metric_task_3[(df_metric_task_3['course_title']==k)\
                                            &(df_metric_task_3['module_number']==l)]['mean_median'].values

            # вычисляем среднее средних значений времени выполнения модулей
            curr_mean = df_metric_task_1[(df_metric_task_1['student_id']==i)\
                                         &(df_metric_task_1['course_title']==k)]['mean'].mean()
            # вычисленные значения между собой
            if summary/len(modules_num) >= curr_mean:
                curr_list.append(0)
            else:
                curr_list.append(1)

        total_list.append(curr_list)
```

Рис. 4.2.22. Вычисление меток успеваемости каждого студента по всем выбранным курсам

```
# записываем вычисленные статусы
status_df['statuses'] = total_list
# приводим к медианным значениям с округлением вниз
status_df['status'] = [math.floor(statistics.median(x)) for x in status_df['statuses']]
status_df
```

	id	courses	statuses	status
0	768c2987a744c51ce64a5993a2a94eaf	[JavaScript с нуля, Анимация интерфейсов, Веб-...	[0, 0, 0]	0
1	03151bc73bdb29fe1be1443c6d83e22f	[UX-дизайн, Анимация интерфейсов, Веб-дизайн P...	[1, 1, 1, 1]	1
2	ed235f47e16da6e83d3f1cb511f38ea6	[Веб-дизайн PRO 2.0, Веб-дизайн с нуля 2.0]	[0, 1]	0
3	59e8681cb7b5c8043ae1aac10c8053ca	[Excel Базовый, Анимация интерфейсов, Веб-диза...	[0, 1, 1]	1
4	c16250079190337fe9074736e33eecb2	[Веб-дизайн PRO 2.0, Веб-дизайн с нуля 2.0]	[0, 1]	0
...	...	...	...	...
9230	a88d8e65143914ccc002c8abbe91324e	[Java-разработчик с нуля]	[1]	1
9231	5b9acd377d0d1b1f2e9e324a44dd0c8a	[Java-разработчик с нуля]	[0]	0
9232	71b5e788516d8e83fb9dc3b5f869dd5b	[Java-разработчик с нуля]	[0]	0
9233	0b77dc9de3ebc312a2ff105bef4b443b	[Java-разработчик с нуля]	[1]	1
9234	b90e440def5b7a643395eed52c02a339	[Java-разработчик с нуля]	[0]	0

9235 rows × 4 columns

Рис. 4.2.23. Вычисление итоговой метки каждого студента

```
# соединяем таблицы
final_df = pd.merge(status_df,\
                    students,\
                    on='id')
# удаляем ненужные и промежуточные атрибуты
final_df.drop(columns=['statuses','courses','id_','birthday'], inplace=True)
final_df
```

	id	status	city	is_working	education	is_married	works_by_profession	has_children	work_places_cnt	age
0	768c2987a744c51ce64a5993a2a94eaf	0	Санкт-Петербург	True	3	False	True	True	5	31
1	03151bc73bdb29fe1be1443c6d83e22f	1	Санкт-Петербург	True	3	True	False	False	6	27
2	ed235f47e16da6e83d3f1cb511f38ea6	0	Москва	True	2	True	False	False	6	30
3	59e8681cb7b5c8043ae1aac10c8053ca	1	Самара	True	3	False	True	False	5	26
4	c16250079190337fe9074736e33eecb2	0	Москва	True	2	False	False	True	5	27
...	...	...	...	...	...	...	...	...	...	...
9230	a88d8e65143914ccc002c8abbe91324e	1	Томск	True	2	True	False	True	4	37
9231	5b9acd377d0d1b1f2e9e324a44dd0c8a	0	Lausanne	True	2	False	False	True	6	47
9232	71b5e788516d8e83fb9dc3b5f869dd5b	0	Омск	False	2	True	True	False	6	35
9233	0b77dc9de3ebc312a2ff105bef4b443b	1	Тимашевск	False	3	False	True	False	5	52
9234	b90e440def5b7a643395eed52c02a339	0	Н.Новгород	True	2	True	True	True	2	45

9235 rows × 10 columns

Рис. 4.2.24. Формирование итогового датафрейма final_df

Таким образом на выходе имеем список студентов с вычисленными метками класса. Те, кто имеет значение status равное «1», не справляется с учебной нагрузкой, а значит нуждается во внимании со стороны преподавателя с целью выяснения обстоятельств.

Разведочный анализ данных

На данном этапе задача заключается в выполнении первоначального исследования данных с целью обнаружения закономерностей, выявления аномалий, проверки гипотез и проверки предположений с помощью сводной статистики и графических представлений.

Будет полезно узнать столбцы и соответствующие им типы данных, а также определить, содержат ли они пустые значения или нет.

```
final_df.shape
```

```
(9235, 10)
```

```
final_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9235 entries, 0 to 9234
Data columns (total 10 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   id                   9235 non-null   object
 1   status               9235 non-null   int64 
 2   city                 9235 non-null   object
 3   is_working           9235 non-null   bool  
 4   education            9235 non-null   int64 
 5   is_married           9235 non-null   bool  
 6   works_by_profession  9235 non-null   bool  
 7   has_children         9235 non-null   bool  
 8   work_places_cnt      9235 non-null   int64 
 9   age                  9235 non-null   int64 
dtypes: bool(4), int64(4), object(2)
memory usage: 541.1+ KB
```

Рис. 4.2.25. Информация о данных датафрейма final_df

Пропущенных значений нет, однако имеются значения типа boolean. Поэтому приведём данные к виду, удобному для дальнейшего анализа. В первую очередь, следует закодировать названия городов. Далее приводим все значения типа boolean к типу integer, где «0» означает False, а «1» – True.

```
# кодируем название городов
cities = final_df['city'].unique()
cities_dict = dict(zip(cities, range(len(cities))))
ml_df = final_df.applymap(lambda s: cities_dict.get(s) if s in cities_dict else s)

# заменяем быстрым способом переменные типа boolean на тип integer
ml_df = ml_df.drop(columns=['id'])*1
ml_df.head()
```

	status	city	is_working	education	is_married	works_by_profession	has_children	work_places_cnt	age
0	0	0	1	3	0	1	1	5	31
1	1	0	1	3	1	0	0	6	27
2	0	1	1	2	1	0	0	6	30
3	1	2	1	3	0	1	0	5	26
4	0	1	1	2	0	0	1	5	27

Рис. 4.2.26. Формирование датафрейма ml_df и приведение категориальных данных к числовым значениям

Применим функцию describe() для получения различной сводной статистики, такой как: количество, среднее значение, стандартное отклонение, минимальное и максимальное значения и квантили данных.

```
ml_df.describe()
```

	status	city	is_working	education	is_married	works_by_profession	has_children	work_places_cnt	age
count	9235.000000	9235.000000	9235.000000	9235.000000	9235.000000	9235.000000	9235.000000	9235.000000	9235.000000
mean	0.711965	340.881646	0.873525	2.447212	0.220249	0.542826	0.535246	5.408230	36.410179
std	0.452872	413.253001	0.332402	0.572359	0.414437	0.498190	0.498783	2.260321	9.480451
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	18.000000
25%	0.000000	1.000000	1.000000	2.000000	0.000000	0.000000	0.000000	4.000000	29.000000
50%	1.000000	139.000000	1.000000	2.000000	0.000000	1.000000	1.000000	6.000000	35.000000
75%	1.000000	602.000000	1.000000	3.000000	0.000000	1.000000	1.000000	7.000000	41.000000
max	1.000000	1485.000000	1.000000	3.000000	1.000000	1.000000	1.000000	9.000000	67.000000

Рис. 4.2.27. Вывод сводной статистики по ml_df

Так как большинство атрибутов являются категориальными (все кроме «work_places_cnt» и «age»), то статистический анализ в данном случае не показал каких-либо явных аномалий в данных.

Рассмотрим количество меток классов.

```
ml_df.status.value_counts()
```

```
1    6575
0    2660
Name: status, dtype: int64
```

Рис. 4.2.28. Подсчёт меток

Получается, что после приведения метрики успеваемости, около 70% студентов отстают по учебной программе в целом. На данном этапе можно сформулировать 2 предположения:

1. Учебные курсы изначально имеют высокую сложность, с чем студенты не справляются.
2. При формировании метрики успеваемости были неправильно высчитаны метки классов.

Пока что предположим, что верно только первое предположение, и продолжим анализ.

Составим корреляционную матрицу всех атрибутов и таргетной переменной.

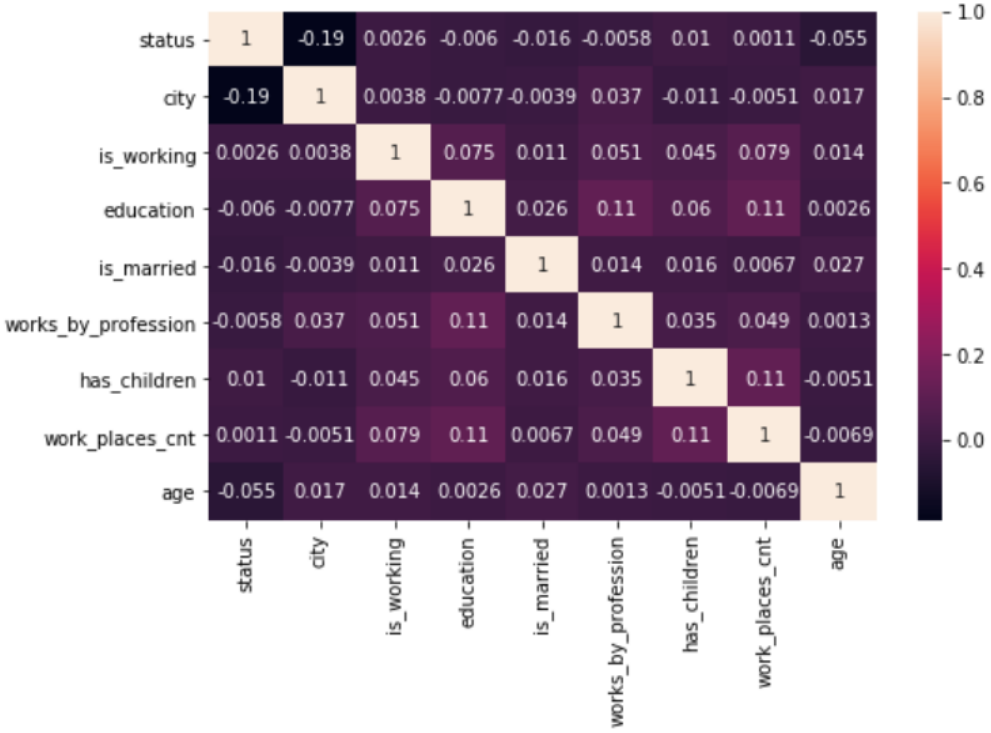

Рис. 4.2.29. Корреляционная матрица ml_df

Отберём те атрибуты, которые имеют корреляцию со статусом по модулю больше либо равную 0.01.

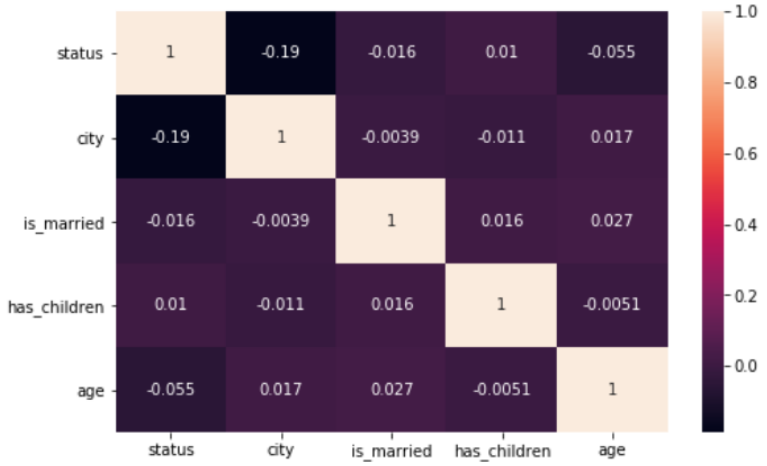

Рис. 4.2.30. Матрица корреляции отобранных атрибутов со статусом

Атрибуты не имеют высокой корреляции между собой, поэтому оставляем их всех для дальнейшего анализа.

Поскольку, из оставшихся атрибутов только возраст не является категориальным, то будем использовать его в качестве основного атрибута для построения графиков.

Построим график подсчёта количества студентов, состоящих в браке, с распределением по возрастам.

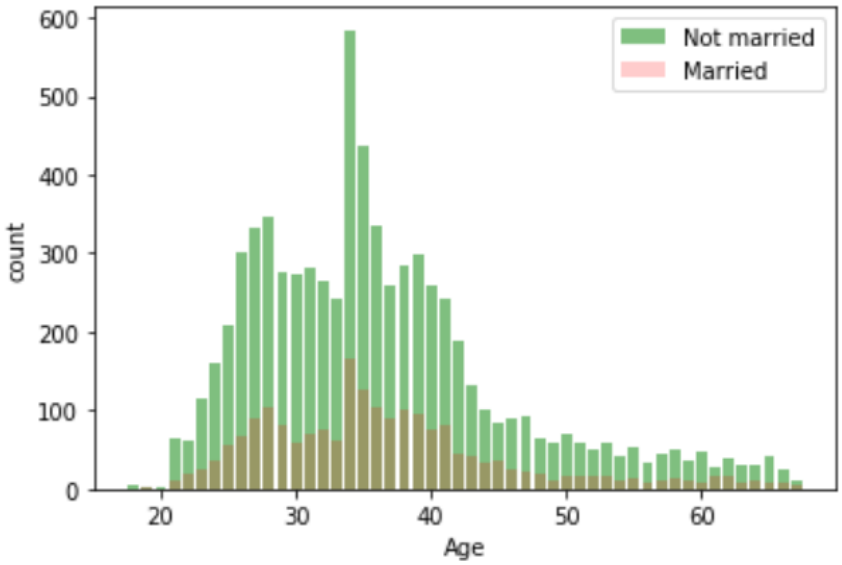

Рис. 4.2.31. График зависимости количества женатых от возраста

При просмотре графика можно отметить, что процентное соотношение по возрастам одинаковое для каждого возраста и составляет примерно 25% женатых к 75% неженатых. Единственное исключение – это студенты до 22 лет, среди которых не нашлось, состоящих в браке. Однако их количество настолько мало, что на графике это сложно заметить.

Построим график подсчёта количества студентов, у которых есть дети, с распределением по возрастам.

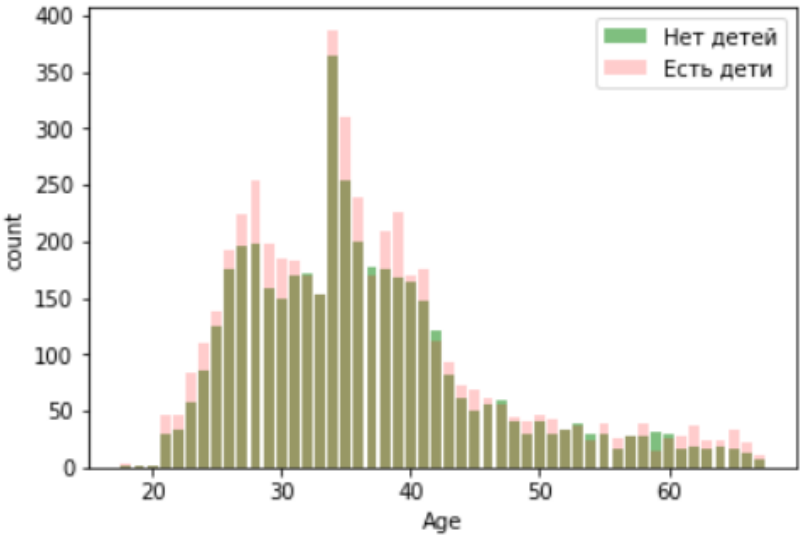

Рис. 4.2.32. График зависимости количества бездетных пользователей от возраста

При просмотре графика можно отметить, что процентное соотношение по возрастам в большинстве случаев одинаковое для каждого возраста и составляет примерно 10% бездетных к 90% пользователям с детьми. Эта разница является крайне несущественной.

Построим график подсчёта количества студентов, у которых есть отставание по учебной программе, с распределением по возрастам.

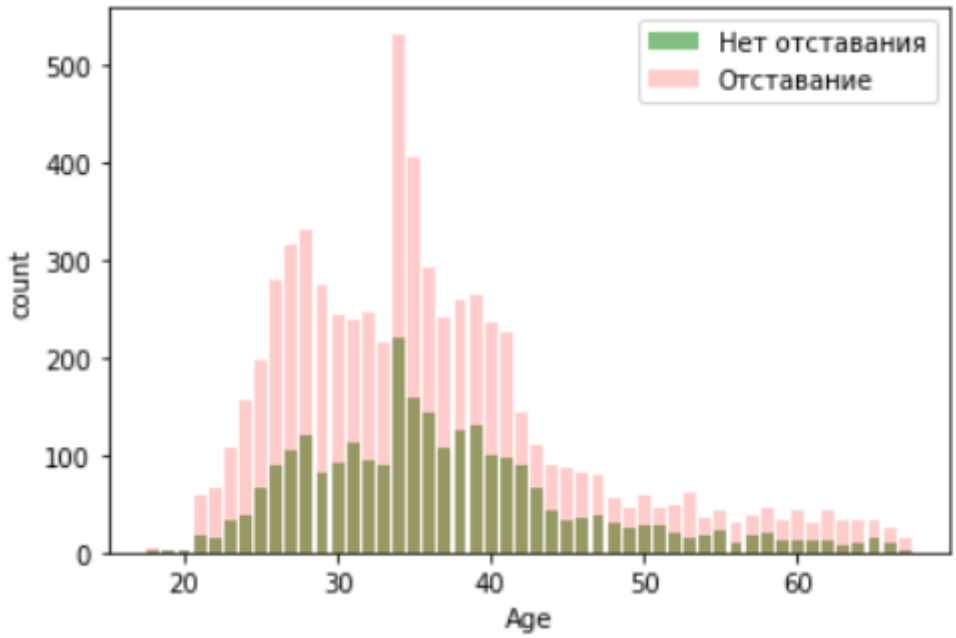

Рис. 4.2.33. График зависимости учебного статуса студента от возраста

При просмотре графика можно отметить, что процентное соотношение по возрастам в большинстве случаев одинаковое для каждого возраста и составляет примерно 40% без отставания по учебной программе к 60% пользователей с отставанием. В остальном, на этом графике не замечены аномалии.

Построим 2 графика зависимости города проживания от возраста.

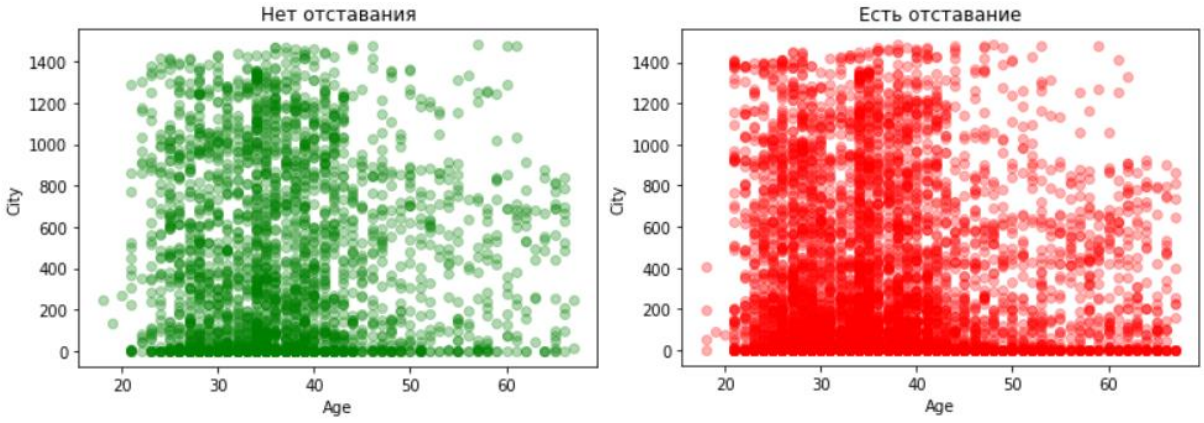

Рис. 4.2.31. Графики зависимости города проживания от возраста

Единственное, что можно подметить, так это большое скопление точек на метках «0» и «1», что соответствует Санкт-Петербургу и Москве. Из этого можно лишь сделать вывод, что большое количество пользователей из конкретного города соответствует численности населения этого города.

В итоге, после проделанного разведочного анализа, не было выявлено ярко выраженных закономерностей и аномалий в данных. Полученный результат порождает вывод о том, что предположение об изначально высокой сложности курсов является ложным. Однако это не обязательно означает, что метрики успеваемости были неправильно посчитаны. Если исходить из проделанной работы, то данные о студентах изначально не были предназначены для проведения такого рода аналитики. Возможно, они даже были созданы случайным образом, так как между ними отсутствует какая-либо явная взаимосвязь. Поэтому построить эффективно работающую модель машинного обучения на этих данных не представляется возможным. Тем не менее построить модели всё равно можно, только ради того, чтобы убедиться, насколько неточно они будут предсказывать новые данные.

Применение методов машинного обучения

На данном этапе задача заключается в построении моделей машинного обучения на основе имеющихся атрибутов и соответствующих им меткам.

Разделим метки классов и атрибуты, а также создадим выборки для обучения и тестирования.

```
# отделяем атрибуты от меток
X = ml_df[['city','is_married', 'has_children', 'age']]
y = ml_df['status']

# разбиваем данные на тренировочную и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,stratify=y)
```

Рис. 4.2.32. Разделение выборки

Обучаем модель логистической регрессии.

```
# создаём экземпляр модели
log_reg = LogisticRegression(max_iter = 50)
# обучаем модель
log_reg.fit(X_train, y_train)
# вычисляем точность для заданных тестовых данных и меток
log_reg.score(X_test, y_test)
```

```
0.7080987440450411
```

```
# предсказываем новые данные
y_pred = log_reg.predict(X_test)
# матрица ошибок
conf_mat=confusion_matrix(y_test,y_pred)
print(conf_mat)
```

```
[[  19  646]
 [  28 1616]]
```

Рис. 4.2.33. Логистическая регрессия

Обучаем модель дерева решений.

```
# стандартизируем функции, удалив среднее значение и масштабируя до единичной дисперсии
X2_train = StandardScaler().fit_transform(X_train)
X2_test = StandardScaler().fit_transform(X_test)
# создаём экземпляр модели
rfc = RandomForestClassifier(n_estimators=50, random_state=0)
# обучаем модель
rfc.fit(X2_train, y_train)
# вычисляем точность для заданных тестовых данных и меток
rfc.score(X2_test, y_test)
```

```
0.6422693806842789
```

```
# предсказываем новые данные
y_pred = rfc.predict(X2_test)
# матрица ошибок
conf_mat=confusion_matrix(y_test,y_pred)
print(conf_mat)
```

```
[[ 166  499]
 [ 327 1317]]
```

Рис. 4.2.34. Дерево решений

Обучаем модель SVM.

```
# создаём экземпляр модели
svm_clf = SVC()
# обучаем модель
svm_clf.fit(X_train, y_train)
# вычисляем точность для заданных тестовых данных и меток
svm_clf.score(X_test, y_test)
```

```
0.7119965352966652
```

```
# предсказываем новые данные
y_pred = svm_clf.predict(X_test)
# матрица ошибок
conf_mat=confusion_matrix(y_test,y_pred)
print(conf_mat)
```

```
[[   0  665]
 [   0 1644]]
```

Рис. 4.2.35. SVM

Обучаем модель CatBoost.

```
# создаём экземпляр модели
cbc = CatBoostClassifier(iterations=100,eval_metric='AUC')
# обучаем модель
cbc.fit(X_train, y_train)
# вычисляем точность для заданных тестовых данных и меток
cbc.score(X_test, y_test)
```

```
82:     total: 190ms    remaining: 39ms
83:     total: 192ms    remaining: 36.6ms
84:     total: 194ms    remaining: 34.3ms
85:     total: 196ms    remaining: 31.9ms
86:     total: 198ms    remaining: 29.6ms
87:     total: 200ms    remaining: 27.3ms
88:     total: 202ms    remaining: 24.9ms
89:     total: 204ms    remaining: 22.6ms
90:     total: 206ms    remaining: 20.3ms
91:     total: 207ms    remaining: 18ms
92:     total: 209ms    remaining: 15.8ms
93:     total: 211ms    remaining: 13.5ms
94:     total: 213ms    remaining: 11.2ms
95:     total: 215ms    remaining: 8.96ms
96:     total: 217ms    remaining: 6.71ms
97:     total: 219ms    remaining: 4.46ms
98:     total: 221ms    remaining: 2.23ms
99:     total: 223ms    remaining: 0us

0.7050671286271113
```

```
# предсказываем новые данные
y_pred = cbc.predict(X_test)
# матрица ошибок
conf_mat=confusion_matrix(y_test,y_pred)
print(conf_mat)
```

```
[[  45  620]
 [  61 1583]]
```

Рис. 4.2.36. CatBoost

4.3 Основные полученные результаты и сравнение с известными подходами

После применения нескольких методов машинного обучения была сформирована сводная таблица с результатами:

Таблица 4.3 – Оценки точности методов машинного обучения

Метод	**TP**	**TN**	**FP**	**FN**	**Accuracy**	**Precision**	**Error rate**
Логистическая регрессия	19	1616	646	28	0,708098744	0,02857143	0,2919013
Дерево решений	166	1317	499	327	0,642269381	0,24962406	0,3577306
SVM	0	1644	655	0	0,715093519	0	0,2849065
CatBoost	45	1583	620	61	0,705067129	0,06766917	0,2949329

Как и предполагалось, качество данных сильно повлияло на статистические характеристики моделей. Точность моделей довольно низкая, а доля ошибок, соответственно, высокая. Возможно, доработав метрику успеваемости, можно было бы немного повысить эффективность моделей, однако это не исправит проблему изначально низкого качества данных.

Заключение

В настоящей дипломной работе рассмотрена задача разработки информационной системы онлайн-портала, предоставляющего возможность прохождения обучающих курсов. Для построения высокоуровневой модели системы предложено использовать язык моделирования UML, являющийся промышленным стандартом. Отдельно рассмотрена задача разработки модуля, анализирующего активность пользователей системы. При выполнении практической части был сделан расчёт нагрузки на потенциальных преподавателей платформы, а также выявлены модули, с которыми у студентов возникает больше всего сложностей. После разработки метрики успеваемости были заданы метки классов, разделяющие пользователей системы на тех, кто справляется с учебной нагрузкой, и на тех, кто не справляется. Далее был проведён разведочный анализ данных с целью нахождения закономерностей между атрибутами и выявления

аномалий. Однако он показал, что связи между атрибутами нет, либо же она крайне слабая ввиду низкого качества изначальных данных. Поэтому было выдвинуто предположение, что методы машинного обучения неудовлетворительно отработают в данном случае. После построения моделей логистической регрессии, дерева решений, SVM и CatBoost полученные результаты, как и ожидалось, подтвердили сделанное предположение.

Список литературы

1. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения. – СПб.: Питер, 2002. – 496 с.
2. Мартин Фаулер. UML. Основы. Краткое руководство по стандартному языку объектного моделирования. 3-е издание. – М.: Символ-Плюс, 2013. – 192 с.
3. Якобсон А., Буч Г., Рамбо Дж. Краткая история UML // Язык UML. Руководство пользователя = The Unified Modeling Language Usere Guide. 2-е. – М.: ДМК Пресс, 2006. – 496 с
4. Rational and UML Partners. UML Specification version 1.1 (OMG document ad/97-08-11) – URL: https://www.omg.org/cgi-bin/doc?ad/97-08-11
5. Кватрани Т. Rational Rose 2000 и UML. Визуальное моделирование: Пер. с англ. – М.: ДМК Пресс, 2013. – 176 с.
6. Леоненков А. В. Нотация и семантика языка UML / Леоненков А. В. 2-е изд., исправ. – М.: Национальный Открытый Университет «ИНТУИТ», 2016. – 205 с. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=429143, свободный.
7. Леоненков А. В. Самоучитель UML 2. – СПб.: БХВ-Петербург, 2007. – 576 с.
8. Джозеф Шмуллер. Освой самостоятельно UML за 24 часа. 3-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 416 с.
9. Рамбо Дж., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка. 2-е издание – СПб.: Питер, 2007. – 544 с.

10. Крэг Ларман. Применение UML 2.0 и шаблонов проектирования. Практическое руководство. 3-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2013. – 736 с.

11. Рашка С. Python и машинное обучение. / пер. с англ. А.В. Логунова. – М.: ДМК Пресс, 2017. – 418 с.

12. Вайолино Боб. Машинное обучение: методы и способы. // Директор информационной службы, 2018. – № 5 – Режим доступа: https://www.osp.ru/cio/2018/05/13054535, свободный. (Дата обращения: 10.04.2022).

13. Rudolph Russell. Machine Learning: Step-by-Step Guide to Implement Machine Learning Algorithms with Python. – CreateSpace Independent Publishing Platform, 2018. – 106 с.

14. Дейтел Пол, Дейтел Харви. Python: Искусственный интеллект, большие данные и облачные вычисления. — СПб.: Питер, 2020. — 864 с.: ил. — (Серия «Для профессионалов»)

15. Вьюгин В.В. Математические основы теории машинного обучения и прогнозирования. – М.: МЦНМО, 2013. — 387 с.

16. Nello Cristianini, John Shawe-Taylor. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. — Cambridge University Press, 2000. – 204 с.

17. Juliana Tolles, William J. Meurer. Logistic Regression: Relating Patient Characteristics to Outcomes. // JAMA, 2016. – Т. 316, № 5. – С. 533-534.

18. Карякина А. А., Мельников А. В. Сравнение моделей прогнозирования оттока клиентов интернет-провайдеров // Машинное обучение и анализ данных, 2017. – Т. 3, № 4. – С. 250-256.

19. Сапегин С. В. Моделирование и оптимизация цифровых сетей передачи данных в САПР информационно-телекоммуникационных систем: дис. … канд. техн. наук: 05.13.12. - Воронеж, 2002. - 162 с.

20. Двоеглазов А.И. Компьютерное моделирование систем управления пучками заряженных частиц: дис. ... канд. физико-математических наук: 05.13.18. – СПб., 2001. – 140 с.
21. Болдырев Е.А. Моделирование и разработка расширяемого программного комплекса для исследований проблемы энергетической безопасности: дис. ... канд. техн. наук: 05.13.18. – Иркутск, 2002. – 152 с.
22. Буч Г., Рамбо Дж., Якобсон И. Введение в UML от создателей языка; Пер. с англ. Н. Мухин. – 2-е изд. – М.: ДМК Пресс, 2010. - 496 с.
23. Конюхова О.В. Графический пользовательский интерфейс для автоматизированных систем раскроя изделий сложной формы: дис. ... канд. техн. наук: 05.13.06. - Орел, 2006. - 170 с.
24. Сотникова А.А. Модульно-структурные средства сбора, обработки и анализа данных в области лечебной профилактики: дис. ... канд. техн. наук: 05.13.17 / Сотникова А. А.; [Место защиты: Пенз. гос. технол. акад.]. – Пенза, 2012. – 169 с.
25. Черняховская Л.Р. Поддержка принятия решений при управлении сложными объектами в критических ситуациях на основе инженерии знаний: дис. ... док. техн. наук: 05.13.01. – Уфа, 2004. – 380 с.
26. Кузнецов М.Б. Трансформация программных моделей и ее применение в технологии MDA: дис. ... канд. физико-математических наук: 05.13.11. – М., 2005. – 136 с.
27. Нуртдинова Э.Э. Интегрированная система информационного обеспечения как фактор повышения эффективности предпринимательской деятельности: дис. ... канд. эконом. наук: 08.00.05. – Уфа, 2015. – 176 с.
28. Торжков И. О. Механизм стратегического управления предприятиями лесного комплекса: дис. ... канд. эконом. наук: 08.00.05 / Торжков И.О.; [Место защиты: Юго-Зап. гос. ун-т]. – Воронеж, 2018. – 211 с.