МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

Заведующий кафедрой
к.т.н., доцент
М.С. Воробьева

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
бакалавра

СИСТЕМА ПРОГНОЗИРОВАНИЯ УСПЕВАЕМОСТИ СТУДЕНТОВ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

02.03.03 Математическое обеспечение и администрирование информационных систем
Профиль «Технологии программирования»

Выполнил работу
студент 4 курса
очной формы обучения
Пахирко Сергей Борисович

Руководитель
Доцент
Иваненко Ольга Александровна

Тюмень
2020

Оглавление

ВВЕДЕНИЕ

Высшее образование является важным этапом в жизни каждого человека, но при поступлении в университет не все абитуриенты могут полностью адаптироваться к меняющимся условиям жизни: с первых дней они погружаются в студенческую действительность, в которой им приходится много работать над домашними заданиями, количество которых постоянно увеличивается. Большинство первокурсников не успевает привыкнуть к новой для них системе, нерационально распределяют свое время, что приводит к накоплению долгов по предметам и последующему отчислению.

С этой проблемой могут столкнуться студенты любого курса. Например, второкурсники подвержены риску быть отчисленными, поскольку во время второго года обучения они начинают относиться к учебе менее серьезно, что приводит к поздно сданным домашним заданиям, увеличению количества пропусков лекций, неудовлетворительным результатам на экзаменах и как итог - отчислению из университета.

Из-за существенного для Вузов процента отчислений студентов по всему миру [2, 18], эта проблема послужила отправной точкой для данной дипломной работы, идея которой состоит в том, чтобы на основе собранных и предобработанных наборов данных об учебной успеваемости дать студентам инструмент для оценивания результатов своей деятельности в перспективе.

Цель данной работы заключается в разработке системы, способную проанализировать учебную успеваемость студентов и получить прогноз результатов сдачи экзаменов и зачетов в предстоящую сессию.

Для достижения этой цели необходимо решить следующие задачи:

- Собрать данные о студентах второго курса направления Математическое обеспечение и администрирование информационных систем за 3 года по всем предметам за 3 семестр.

- Изучить методы машинного обучения и анализа данных. Провести исследования применимости методов классификации и регрессии для прогнозирования успеваемости.
- Спроектировать базу данных для хранения информации о студентах.
- Разработать серверную часть системы для хранения, предобработки, анализа данных и для взаимодействия с базой данных.
- Разработать мобильное приложение для просмотра успеваемости студентов.

ГЛАВА 1. ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

1.1. Научные исследования

Проблема успеваемости студентов, является одной из ключевых для многих университетов мира. Проводится большое число научных исследований, целью которых является выявление факторов, влияющих на учебный процесс студентов и успеваемость студентов в будущем.

Например, в статье «EMT: Ensemble Meta-Based Tree Model for Predicting Student Performance» авторы данной научной работы проводят детальное исследование учебных данных за прошлые семестры, для выявления ключевых признаков, также анализируют различные методы машинного обучения для прогнозирования учебной успеваемости студентов. Они используют как одиночные методы, так и комбинации алгоритмов, каждый из которых обучается независимо друг от друга, а при классификации результаты определяются путем голосования. Проведя несколько тестов, исследователи определили, что наибольшую эффективность показали комбинации алгоритмов, особенно NBTree и Adaboost_J48, что позволило им получить точность прогнозируемой успеваемости студентов в районе 98.5% [1].

В статье «Prediction of Student's performance by modelling small dataset size» исследуют проблему студенческой успеваемости, и автор ставит перед собой цель – классифицировать студентов, у которых есть риск отчисления. В статье подробно описаны все этапы исследования данных, такие как: обработка категориальных признаков, заполнение пропущенных значений и нормализация. Ключевая особенность данной работы в том, что автор исследует работу нескольких алгоритмов машинного обучения (LDA, KNN, SVM, NB, MLP-ANN) на небольшом объеме данных (50 студентов). После сравнения алгоритмов классификации, автор статьи пришел к выводу, что при правильном подходе к исследованию данных даже на не большом объеме данных можно показать хорошие результаты. Из всех рассмотренных методов высокую точность показали KNN и LDA (точность классификации студентов составила 74-79%) [11].

В России также проводились подобные исследования. В статье «Нейросетевая модель прогнозирования группы риска по успеваемости студентов первого курса» авторы из Пермского государственного национального исследовательского университета исследуют влияние довузовских факторов (учебное заведение, баллы ЕГЭ, город образовательного учреждения и др.) на успеваемость студентов в первом году обучения. Для построения нейронных сетей использовали STATISTICA Automated Neural Networks. Объем имеющихся данных для составлял 274 объекта, что может быть недостаточно для обучения нейросетевых моделей, поэтому авторы использовали метод создания дополнительных обучающих данных из имеющихся, что позволило дополнить данные до 548 объектов. Точность прогнозирования составила 80%. В заключении исследования, авторы пришли к выводу, что для получения более точного прогноза им необходимо привлечь дополнительные факторы, связанные с психофизическим состоянием студентов [20].

1.2. Информационные системы для мониторинга успеваемости

Множество университетов на основе данных исследований создают различные системы, призванные наглядно показать студентам их уровень обучения и улучшить студенческую успеваемость.

Так, например, английский Университет Ноттингем Трент создал систему для мониторинга не только студенческой успеваемости, но и вовлеченности студентов в учебный процесс, которая была подробно описана в статье «CASE STUDY I: Predictive analytics at Nottingham Trent University». Данная система разрабатывалась для снижения показателя отсева студентов, улучшения успеваемости, анализа активности студента в университетской деятельности. Основной целью данного проекта было «обеспечить студентов инструментом, который мотивировал бы их» [4]. По прошествии трех лет после внедрения системы, 72% первокурсников, сообщили, что использование данного сервиса способствовало улучшению их успеваемости.

Интересен опыт другого зарубежного вуза. Университет Пердью в США разработал систему предиктивной аналитики, которая на основе данных об академической успеваемости студентов, их активности в цифровой учебной среде и демографических данных рассчитывает уровень риска отсева для каждого студента, после чего данная информация отправляется куратору группы и самому студенту. Благодаря такой интерактивной системе университету удалось улучшить результаты обучения и снизить показатели отсева.

Опираясь на опыт зарубежных вузов, можно предположить, что реализация подобной системы может повысить уровень успеваемости российских студентов и уменьшить процент отчислений из университетов.

ГЛАВА 2. ИССЛЕДОВАНИЕ И РЕАЛИЗАЦИЯ МЕТОДОВ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

2.1. Постановка задачи

Для анализа успеваемости студентов будут использоваться различные методы и подходы из такой области интеллектуального анализа данных, как машинное обучение.

Методы машинного обучения – это алгоритмы, использующие математические методы для нахождения паттернов в больших объемах данных. Они применяются для решения различных видов задач, таких как: классификация, регрессия, кластеризация [9].

Дадим формальное определение задач машинного обучения:

Пусть имеется множество описаний объектов - X и конечное множество допустимых ответов - Y. Существует некое отображение $y^* = X \to Y$, значения функции которой $y^*(x_i)$ известны только на конечном подмножестве объектов $\{x_1, \dots, x_l\} \subset X$. Совокупность пар $X^l = (x_i, y_i)^{\,l}{}_{i=1}$ называется обучающей выборкой. Задача состоит в том, чтобы построить алгоритм $a = X \to Y$, который приближал бы целевую функцию $y^*(x)$ на всем множестве X.

Этап обучения модели является наиболее важным и сложным, и он состоит из поиска параметров модели, благодаря которым достигается приемлемое значение функционала качества.

$$Q(a, X^l) = \frac{1}{l}\sum_{i=1}^{l} L(a, x_i) \qquad (2.1)$$

где $Q(a, X^l)$ - функционал качества алгоритма a на выборке X^l,

$L(a, x)$ - функция потерь, характеризующая величину ошибки алгоритма a на выборке X^l.

Классический метод обучения заключается в том, чтобы найти в заданной модели A такой алгоритм a, на котором будет достигаться минимум функционала качества:

$$Q(a, X^l) \to min_{\,a \in A} \qquad (2.2)$$

Прогноз студенческой успеваемости будет рассматриваться для трех случаев:

1) Множество допустимых значений по предмету «Объектно-ориентированное программирование» (ООП) – {зачёт / не зачёт};
2) Множество допустимых значений по результатам семестра – {Отчислен/не отчислен};
3) Вещественное число, характеризующее приблизительную оценку за экзамен по предмету «Структуры и алгоритмы компьютерной обработки данных» (СИАКОД).

В первых двух случаях необходимо решить задачу бинарной классификации, где множество допустимых значений принимает вид $Y = \{0, 1\}$ и множество объектов X разбивается на классы $K_y = \{x \in X: y^*(x) = y\}$. Алгоритм $a(x)$ должен дать ответ на вопрос «x принадлежит первому или второму классу?»

Иными словами, задача классификации — это определение на основе входных признаков, к какому классу из уже известных относится новое наблюдение.

Для третьего случая необходимо решить задачу регрессии, где множество допустимых значений $Y \in [2,5]$ и результатом работы алгоритма $a(x)$ будет являться вещественное число $y^*(x) \in [2,5]$, приближенное к истинной оценке по предмету.

Для построения систем с машинным обучением зачастую используются специальные методологии по исследованию и работе с данными, которые позволяют оценить их с разных сторон. Одна из самых широко используемых методологий на сегодняшний день - CRISP-DM (CRoss-Industry Standard Process for Data Mining – межотраслевой стандартный процесс для исследования данных) [15].

CRISP-DM – это проверенная в промышленности методология по исследованию данных [13].

Модель жизненного цикла CRISP-DM состоит из шести фаз.

1. Понимание бизнес-целей (Выявление важных факторов, влияющих на успеваемость студентов).
2. Начальное изучение данных (Проверка собранных данных, построение гипотез, изучение корреляций между данными).
3. Предобработка данных (Очистка данных от различных «шумов», ошибочных значений и их подготовка в удобный для машины вид).
4. Моделирование (Построение моделей машинного обучения).
5. Оценка (Оценка результатов моделей).
6. Внедрение (Внедрение успешных моделей).

Основное преимущество методологии в том, что она циклична, что позволяет пересматривать ранние этапы с учетом полученных результатов.

2.2. Сбор и подготовка данных

Для алгоритмов машинного обучения важную роль играют данные, т.к. на их основе происходит обучение и делаются какие-либо выводы. Чем больше качественных данных будет получено моделью, тем лучше будут её результаты [7].

Данные могут быть разного вида: количественные, тексты, изображения. В нашем случае — это материалы успеваемости студентов (баллы за контрольные недели, количество пропусков лекций, баллы за Единый Государственный Экзамен). Для прогнозирования успеваемости были собраны сведения о студентах 2 курса направления Математическое обеспечение и администрирование информационных систем за последние 3 года (группы 164, 174, 184), общее число студентов составило 143 человека.

С участием преподавателей и учебной части Института математики и компьютерных наук было собрано 52 файла формата xlsx:

- Баллы за контрольные недели
- Ведомости по предметам
- Файлы с итоговыми результатами за семестр

- Журналы от преподавателей
- Данные о баллах ЕГЭ

После просмотра и отбора данных, было подготовлено три итоговых набора данных формата xlsx:

1. Данные по предмету ООП.
2. Данные студентов за 3 семестр обучения по предметам:
    a. Объектно-ориентированное программирование.
    b. Структуры и алгоритмы компьютерной обработки данных.
    c. Дифференциальная геометрия.
    d. Дифференциальные уравнения.
    e. Архитектура вычислительных систем и компьютерных сетей.
3. Данные по предмету СИАКОД.

Собранные материалы содержат сведения об учебной успеваемости и персональную информацию о студентах, такую как: ФИО, пол, группа, баллы ЕГЭ, тип обучения студента (бюджетная/платная основа обучения). Пример данных приведен в таблице 2.1.

Таблица 2.1

Пример персональных данных

ФИО	Пол	Группа	Баллы за ЕГЭ	Тип обучения
-	Мужской	174-2	193	Бюджетная основа
-	Женский	174-2	239	Бюджетная основа
-	Мужской	174-2	209	Бюджетная основа
-	Мужской	184-2	211	Платная основа
-	Мужской	184-2	235	Бюджетная основа

В таблице 2.2 представлена часть набора данных по предмету «Объектно-ориентированное программирование», где троеточием обозначаются пропущенные колонки.

Таблица 2.2

Пример данных по предмету ООП

Баллы за первую контрольную неделю	Пропущенные лекции за первую контрольную неделю	Посещенные лекции за первую контрольную неделю	Баллы за вторую контрольную неделю	…	Результат
22	0	6	42		Зачтено
19	2	5	0		Зачтено
10	0	6	4		Зачтено
2	3	3	0		Не зачтено
8	1	5	0		Не зачтено
11	0	6	25		Зачтено

Также, в таблице 2.3, представлен набор данных, содержащий баллы за контрольные недели (для примера в таблице представлена только первая контрольная неделя) для прогнозирования итогов по всему семестру.

Таблица 2.3

Пример данных по предметам 3 семестра

Диф. Ур. Кр.н. 1	СИАКОД Кр.н. 1	Диф. Геом. Кр.н. 1	ООП Кр.н. 1	АВСИКС Кр.н. 1	…	Итог
4	14	20	22	20		Не отчислен
0	12	0	19	0		Отчислен
8	12	5	10	11		Не отчислен
2	0	5	2	5		Не отчислен
9	0	5	8	8		Не отчислен

В файле и базе данных, названия признаков указаны на английском языке. В таблице 2.4 приведены типы признаков и их описание на русском языке.

Таблица 2.4

Описание признаков

Название признака	Описание признака	Тип
points_week_1 points_week_2 points_week_3	Баллы за первую, вторую и третью контрольную неделю соответственно	Число
missed_week_1 missed_week_2 missed_week_3	Количество пропущенных лекций за первую, вторую и третью контрольную неделю соответственно (Известно только по предмету ООП)	Число
visited_week_1 visited_week_2 visited_week_3	Количество посещённых лекций за первую, вторую и третью контрольную неделю соответственно (Известно только по предмету ООП)	Число
semester_mark	Оценка по предмету по набранным баллам в течение семестра	Для первого и второго случая – бинарный тип. Для третьего случая – целое число
group	Номер группы студента	Строка
gender	Пол студента	Бинарный
type_study	Тип обучения студента	Бинарный
USE	Баллы за ЕГЭ	Число
exam_mark	Итоговый результат	Для первого и второго случая – бинарный тип. Для третьего случая – вещественное число

Помимо сбора данных, важным этапом в разработке систем с машинным обучением является их предобработка, т.к. качество данных может серьезно влиять на точность модели. Предобработка включает такие задачи, как: поиск взаимосвязей между данными, нормализация и дискретизация данных, обработка экстремальных значений, извлечение новых признаков [17].

В рамках данной работы обработка данных включает в себя такие пункты, как:

1) Обработка пропущенных значений.

   Если некоторые записи имеют поля с пустыми значениями, их необходимо обработать для правильной работы большинства моделей. Есть несколько способов для борьбы с пропущенными значениями.

   Их можно заполнить средним или медианным значением по этому полю, использовать константу (например, если возраст у студента не известен, то по умолчанию он будет равен 18) или использовать методы регрессии. Если количество пропущенных значений невелико, можно удалить их из выборки.

   В подготовленных наборах данных присутствовали пропущенные значения в колонке «Баллы ЕГЭ». Пропущенные данные были заполнены средним значением баллов по группам.

2) Нормализация данных.

   Нормализация данных необходима для корректной работы методов машинного обучения, т.к. значения признаков могут отличаться друг от друга в несколько раз. Поэтому необходимо привести значения к одному диапазону, зачастую $[0, 1]$. Для всех числовых атрибутов из подготовленных наборов данных была выполнена нормализация методом «минимакс» (пример нормализованных данных представлен в таблице 2.5)

   $$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}, \qquad (2.3)$$

   где X_{min} – минимальное значение вектора X,

   X_{max} – максимальное значение вектора.

Таблица 2.5

Пример нормализации данных

Номер студента	Баллы за вторую контрольную неделю	Баллы за ЕГЭ	Баллы за вторую контрольную неделю (после нормализации)	Баллы за ЕГЭ (после нормализации)
1	42	216	0,84	0,46
2	0	213	0	0,44
3	4	193	0,08	0,26
4	0	239	0	0,66
5	0	209	0	0,40

3) Было произведено преобразование категориальных признаков (пол, тип обучения) в бинарные.

Код методов по предобработке данных находится в приложении 1.

2.3. Анализ данных

После подготовки данных обязательным этапом является их анализ. Так, с помощью матрицы корреляций был проведен анализ степени взаимосвязи признаков.

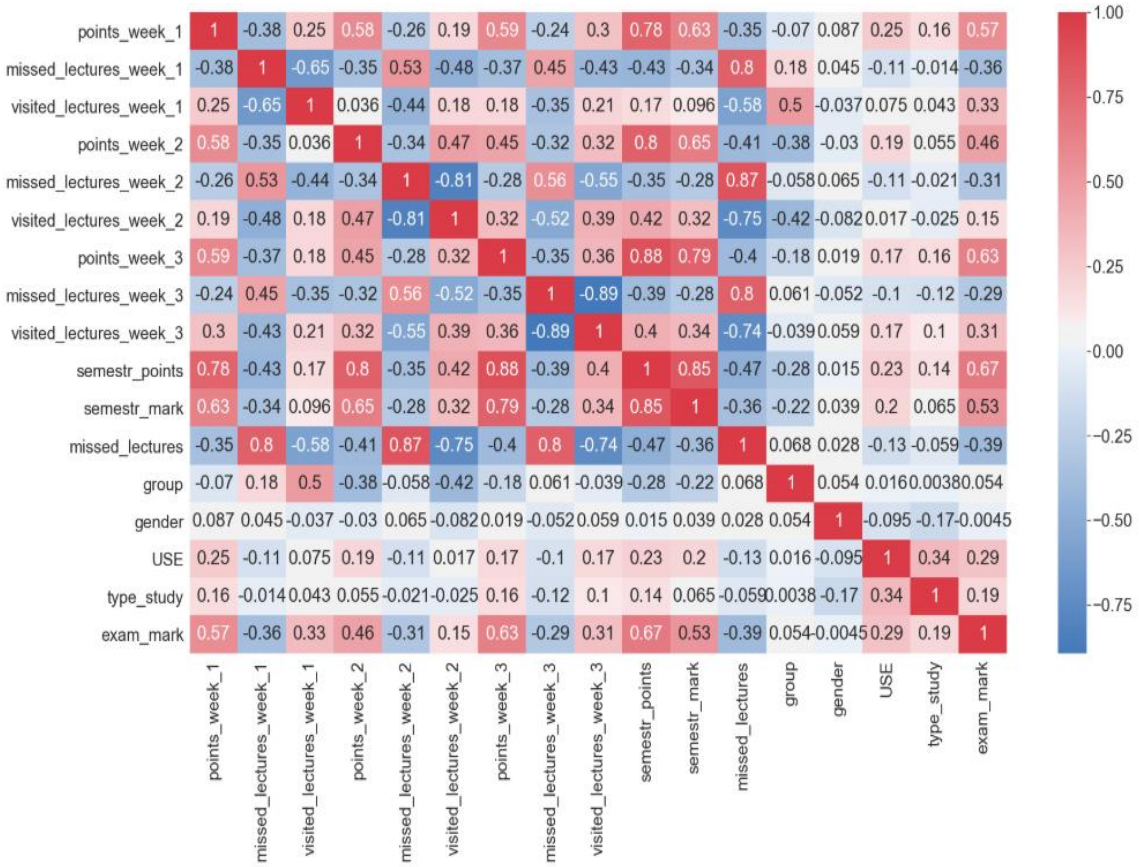

Рис. 2.1. Матрица корреляций признаков по предмету ООП

Как видно на рисунке 2.1., у признаков «gender» и «group» (описание характеристик приведено в таблице 2.4) корреляция с целевой переменной практически равна 0, и можно сделать вывод, что данные признаки не оказывают сильного влияния, и их можно удалить.

Одной из проблем, которое могут возникнуть при решении задач машинного обучения – это проблема несбалансированности данных. Так как «успешных» студентов (которые закрывают все зачеты и экзамены во время сессии) всегда больше, чем «неуспешных» (которые имеют проблемы с учебой).

Несбалансированные наборы данных – это наборы данных, в которых записей одного или нескольких классов в разы больше (или меньше), чем записей других.

Действительно, как показано на рисунке 2.2., объектов одного класса гораздо больше, чем другого (не отчисленных студентов 72 человека, отчисленных 17).

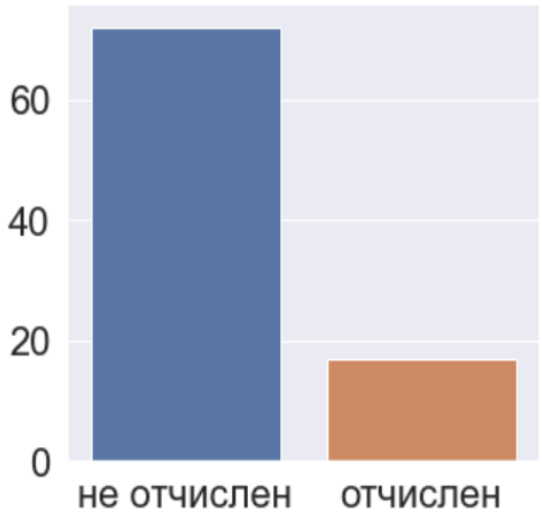

Рис. 2.2. Сравнение числа отчисленных
и не отчисленных студентов

Существуют различные способы по устранению дисбаланса классов, такие как [14]:

- Использовать одинаковое количество данных из каждого класса.
- Воспользоваться повторяющимися записями из меньших классов, для увеличения их количества.
- Использовать алгоритмы для искусственного увеличения данных (например, SMOTE).

Для работы алгоритмов машинного обучения критически важным является качество и количество данных. Т.к. размер датасета студентов небольшой (143 записи), может возникать проблема переобучения: когда модель хорошо выявляет закономерности во время обучения, но при добавлении новых данных, с которыми она не сталкивалась, её точность существенно падает.

Решить данную проблему можно несколькими способами. Наиболее правильным является увеличение количества и качества данных для обучения, т.к. это позволит выявить большинство возможных закономерностей. Другим вариантом решения проблемы является использование методов машинного обучения, которые не сильно подстраиваются под данные.

2.4. Используемые методы классификации

Для решения задачи бинарной классификации было решено использовать следующие алгоритмы:

1. KNN (k-ближайших соседей)
2. Support Vector Machine (SVM) – метод опорных векторов
3. Logistic Regression (Логистическая регрессия)
4. Random Forest (Случайный лес)

1) Метод KNN (k-ближайших соседей) – является одним из самых простых метрических алгоритмов классификации, основанный на вычислении оценки сходства между ближайшими объектами.

Будем относить объект u к тому классу, элементов которого окажется больше среди k ближайших соседей (2.4).

$$a(u; X^l, k) = argmax_{y \epsilon Y} \sum_{i=1}^{k} [y_u^{(i)} = y], \qquad (2.4)$$

где k для бинарной классификации выбирается нечетное.

2) SVM – метод опорных векторов, применяющийся для решения задач классификации и регрессии, который заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом.

Главная цель SVM как классификатора - найти уравнение разделяющей гиперплоскости $w_1 x_1 + w_2 x_2 + ... + w_n x_n + w_0 = 0$ в пространстве R^n, которая разделила бы два класса неким оптимальным образом.

$$F(x) = sign(w^T x - b), \qquad (2.5)$$

где $F(x)$ - функция преобразования входных данных x в метку класса Y, а

$$w^T = (w_1, w_2, \ldots, w_n);\ b = -w_0, \qquad (2.6)$$

где w – веса атрибутов.

Все объекты, попадающие по одну сторону, будут классифицироваться как первый класс, а по другую - как второй класс.

Веса w и b настраиваются таким образом, чтобы объекты классов лежали как можно дальше от разделяющей гиперплоскости, другими словами, необходимо максимизировать зазор между гиперплоскостью и объектами класса.

3) Logistic Regression – метод логистической регрессия представляет собой линейную модель бинарной классификации.

В основе логистической регрессии лежит вероятностная модель и важным понятием для нее является отношение шансов - $\frac{p}{1-p}$, где p — это вероятность наступления положительного события.

Затем, мы можем определить логарифмическую функцию отношения шансов, которая принимает входные значения в диапазоне от 0 до 1 и трансформирует их в вещественные числа, что позволяет использовать их для выражения линейной связи между значениями признаков и логарифмами отношения шансов (2.6).

$$\log\frac{p}{1-p} = \sum_{i=0}^{n} w_i x_i = w_0 x_0 + w_1 x_1 + \ldots + w_n x_n, \qquad (2.6)$$

Для того, чтобы предсказать вероятность принадлежности к определенному классу, необходимо воспользоваться обратной формой логарифмической функции, которую еще называют логистической функцией или сигмоидой (2.7).

$$\varphi(z) = \frac{1}{1+e^{-z}}, \qquad (2.7)$$

где z - линейная комбинация весов и признаков образца [18],

$$z = w^T x = w_0 x_0 + w_1 x_1 + \ldots + w_n x_n. \qquad (2.8)$$

4) Random Forest - алгоритм машинного обучения, в основе которого лежит построение ансамбля решающих деревьев.

Решающие деревья (Decision Tree) - представляют собой иерархические древовидные структуры, состоящие из решающих правил вида «Если …, то...», формирующихся в процессе обучения.

Алгоритм случайного леса использует две ключевые концепции:

- Образцы из набора данных выбираются случайно.

  В процессе обучения, каждое дерево случайного леса учится на случайном наборе данных из обучающей выборки. Выборка образцов происходит с возмещением (bootstrapping), что дает возможность повторного использования этих данных другим деревьям.
- При разделении узлов выбираются случайные наборы параметров.

  Случайный лес обучает каждое дерево на отдельной выборке данных, разделяя узлы в каждом дереве с использованием ограниченного набора параметров.

Итоговый прогноз делается путем усреднения прогнозов от всех деревьев [10].

2.5. Метрики оценивания качества классификации

Чтобы понять, какой алгоритм показал наилучшие результаты нужно оценить качество работы алгоритмов [2].

Наиболее часто используемый способ оценки качества классификации - доля правильных ответов (accuracy).

$$accuracy(a, X) = \frac{1}{l}\sum_{i=1}^{l}[a(x_i) = y_i], \qquad (2.9)$$

Однако эта метрика имеет серьезной недостаток:

При несбалансированности данных, где, объектов одного класса будет 950, а объектов другого класса 50, и если классификатор отнесёт все объекты к большему классу, то доля правильно классифицированных объектов будет равняться 0.95. Это будет обозначать то, что нарушен баланс классов и что полученная модель не является приемлемой для решения задачи.

Для соотношения между результатами работы алгоритма и истинным ответом удобно использовать матрицу ошибок (таблица 2.6):

Таблица 2.6

Матрица ошибок

	y = 1	y = -1
a(x) = 1	True Positive (TP)	False Positive (FP)
a(x) = -1	False Negative (FN)	True Negative (TN)

Если алгоритм относит ответ к классу +1, то говорят, что алгоритм срабатывает. Если алгоритм сработал, и ответ действительно относится к классу +1, имеет место верное срабатывание (true positive), а если объект относится к классу -1, имеет место ложное срабатывание (false positive).

Если алгоритм дает ответ -1, говорят, что он пропускает объект. Если был пропущен объект класса +1, то это ложный пропуск (false negative). Если же алгоритм пропускает объект класса -1 — это истинный пропуск (true negative).

Таким образом, существует два вида ошибок: ложное срабатывание и ложные пропуски. И для каждой ошибки нужна своя метрика качества.

Введем две метрики:

1) Точность (precision) - данная метрика показывает, насколько можно доверять классификатору в случае срабатывания.

$$precision(a, X) = \frac{TP}{TP + FP}, \qquad (2.10)$$

2) Полнота (recall) - на какой доле истинных объектов первого класса алгоритм срабатывает.

$$recall(a, X) = \frac{TP}{TP + FN}, \qquad (2.11)$$

Также существует еще одна метрика, которая позволяет объединить точность и полноту - гармоническое среднее (F-мера):

$$F = (1 + \beta^2)\frac{precision * recall}{\beta^2 * precision + recall}, \qquad (2.12)$$

где β - параметр, который позволяет регулировать влияние точности и полноты.

2.6. Используемые методы регрессии

Для прогнозирования оценок по предмету «Структуры и алгоритмы компьютерной обработки данных» (СИАКОД) будет использоваться метод линейной регрессии (Linear Regression) и его модификации.

Алгоритм линейной регрессии выглядит следующим образом:

$$a(x) = w_0 + \sum_{j=1}^{d} w_j x^j, \qquad (2.13)$$

где w_0 - свободный коэффициент, x^j признаки, а w_j - их веса.

Функция потерь для линейной модели часто выбирается в виде квадрата отклонения - $(a(x) - y)^2$.

Функционал качества, который называется среднеквадратичной ошибкой алгоритма, задается следующим образом:

$$Q(a, X^l) = \frac{1}{l}\sum_{i=1}^{l}(a(x_i) - y_i)^2, \qquad (2.14)$$

Так как необходимо минимизировать функционал качества для обучения модели, нужно воспользоваться численными методами оптимизации, а именно методом градиентного спуска. С помощью данного метода будут подобраны оптимальные веса для алгоритма.

В случае, когда модель получается слишком сложной и недостаточно данных для точного определения параметров, модель может легко переобучиться. Решить эту проблему можно либо упростив модель, либо использовав регуляризацию – ограничение значений весов модели.

Существуют такие типы регуляризации как:

L_2-регуляризатор (ridge-regression):

$$Q(a, X^l) = \frac{1}{l}\sum_{i=1}^{l}(a(x_i) - y_i)^2 + \lambda\sum_{j=1}^{d} w_j^2 \rightarrow min_w, \qquad (2.15)$$

L_1-регуляризатор (lasso-regression):

$$Q(a, X^l) = \frac{1}{l}\sum_{i=1}^{l}(a(x_i) - y_i)^2 + \lambda\sum_{j=1}^{d}|w_j| \rightarrow min_w, \qquad (2.16)$$

При использовании L_2-регуляризации значения весов сглаживаются, а при использовании L_1-регуляризации веса признаков, обладающих низкой предсказательной способностью, откидываются, т.к. они становятся равны 0.

2.7. Метрики оценивания качества регрессионных моделей

Первая метрика, которая используется не только для оценивания модели, но и в качестве функции ошибки, это среднеквадратичная ошибка (2.17):

$$MSE\ (a, X^{\,l})\ = \frac{1}{l}\sum_{i=1}^{l}(a(x_i) - y_i)^2, \qquad (2.17)$$

Данную метрику сложно интерпретировать, поэтому для оценки качества регрессионной модели зачастую используется коэффициент детерминации [9]:

$$R^2(a, X^{\,l})\ =\ 1\ -\ \frac{\sum_{i=1}^{l}(a(x_i) - y_i)^2}{\sum_{i=1}^{l}(y_i - \bar{y})^2}, \qquad (2.18)$$

где $\bar{y} = \frac{1}{l}\sum_{i=1}^{l} y_i$. (2.19)

Данный коэффициент позволяет легко интерпретировать значение среднеквадратичной ошибки. Коэффициент детерминации для модели принимает значение $0 \leq R^2 \leq 1$. Модель считается приемлемой, если её коэффициент детерминации не меньше 50%. Чем ближе коэффициент детерминации к 1, тем лучше модель описывается целевую переменную. Если коэффициент выше 80%, то модель можно признать хорошей [12].

2.8. Результаты исследовательской работы

После реализации выбранных алгоритмов был проведен анализ моделей при решении задачи классификации по предмету Объектно-ориентированное программирование. Результаты сравнения моделей представлены в таблице 2.7:

Таблица 2.7

Результаты классификации по предмету ООП

	accuracy	precision	recall	f1-score
KNN	0.7	1.0	0.63	0.77
Logistic Regression	0.75	0.97	0.72	0.83
SVM	0.72	0.97	0.67	0.79
Random Forest	0.68	0.96	0.63	0.76

Качество моделей оценивалось с помощью метрики f1-score, и наилучшие результат показал метод Логистической регрессии (Logistic Regression).

Было проведено сравнение работы моделей и при прогнозировании характеристики отчислен / не отчислен (таблица 2.8):

Таблица 2.8

Результаты классификации результатов обучения по итогам 3 семестра

	accuracy	precision	recall	f1-score
KNN	0.88	0.9	0.88	0.89
Logistic Regression	0.81	0.83	0.81	0.82
SVM	0.85	0.88	0.85	0.86
Random Forest	0.85	0.88	0.85	0.86

Хорошие результаты показал метод К-ближайших соседей.

В таблице 2.9, приведено сравнение результатов работы методов регрессии для прогнозирования оценок на данных по предмету СИАКОД:

Таблица 2.9

Результаты регрессионных методов по предмету СИАКОД

	Linear Regression	Lasso Regression	Ridge Regression
MSE	0.3934	0.4606	0.393
R^2	0.3589	0.2493	0.3595

Более наглядно оценить получившиеся результаты регрессии можно на спрогнозированных оценках, которые сравниваются с реальными в таблице 2.10:

Таблица 2.10

Спрогнозированные оценки для 8 студентов.

Номер студента	Истинная оценка	Спрогнозированная оценка
1	5	3.99
2	3	3.24
3	4	3.53
4	4	4.03
5	4	3.95
6	3	2.54
7	2	2.86
8	5	4.41

ГЛАВА 3. РАЗРАБОТКА СИСТЕМЫ

3.1. Описание архитектуры системы

Система по прогнозированию успеваемости студентов представляет собой клиент-серверное приложение, построенное на основе архитектурного стиля REST для взаимодействия компонентов распределенного приложения в сети.

Клиентское приложение разработано для мобильных устройств с операционной системой Android, а серверная часть системы состоит из трех компонент (архитектура системы приведена на рисунке 3.1):

- Серверное приложение - основной компонент серверной системы, предназначенный для обработки запросов клиентов, взаимодействия с базой данных и использования сервиса для прогнозирования студенческой успеваемости.
- База данных - компонент системы, отвечающий за хранение данных результатов обучения студентов.
- Сервис с машинным обучением - часть серверной системы, которая используется для прогнозирования академических результатов студентов.

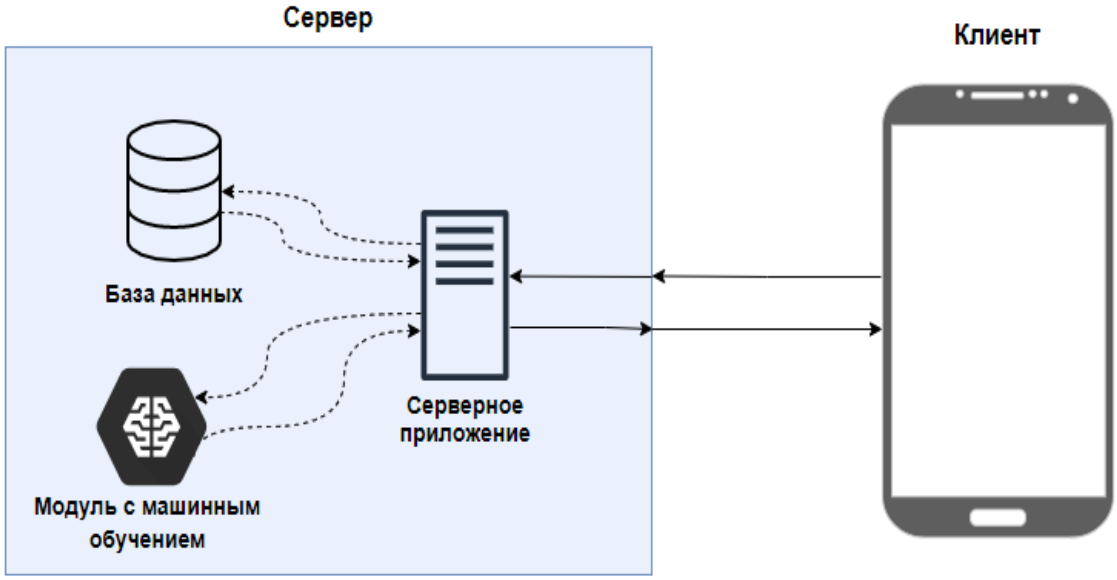

Рис. 3.1. Архитектура системы

Для полноценной работы серверной части системы в сети Ethernet её необходимо развернуть на специализированных платформах, предоставляющих

удаленные сервера. В качестве платформы для размещения был выбран сервис Digital Ocean, который позволяет создавать виртуальные машины на облачных серверах компании для реализации различных видов задач своих клиентов.

3.2. Проектирование базы данных

Собранные данные включают информацию о студентах, их учебных предметах и успеваемости по ним. Помимо имеющихся данных о студентах, необходимо хранить пользователей системы, разделять их на два вида с разными правами: студент и преподаватель.

Для хранения информации о студентах и их академической успеваемости была выбрана нереляционная база данных MongoDB.

Базы данных разделяются на два основных вида: реляционные и нереляционные.

Реляционная база данных — это база данных, в которой вся информация об объектах хранится в виде таблиц и таблицы имеют заранее предопределенные связи. Второй вид баз данных — нереляционные, основное их отличие в том, что вместо таблиц данные хранятся в документах типа JSON, что позволяет иметь более гибкую структуру [5].

Данное свойство является большим плюсом, т.к. данные некоторых предметов могут отличаться наличием дополнительных полей, например, пропущенные лекции у каждого студента. Помимо сведений о студентах и их академической успеваемости, необходимо хранить информацию о пользователях системы, так как разные пользователи имеют разные поля в базе данных: студенты содержат данные по успеваемости, у преподавателей они отсутствуют.

Концепция документоориентированной базы данных в случае добавления новых признаков позволит без труда изменить её структуру, не создавая дополнительные таблицы и связи между ними, что в случае масштабирования системы позволит в короткое время видоизменить структуру базы данных.

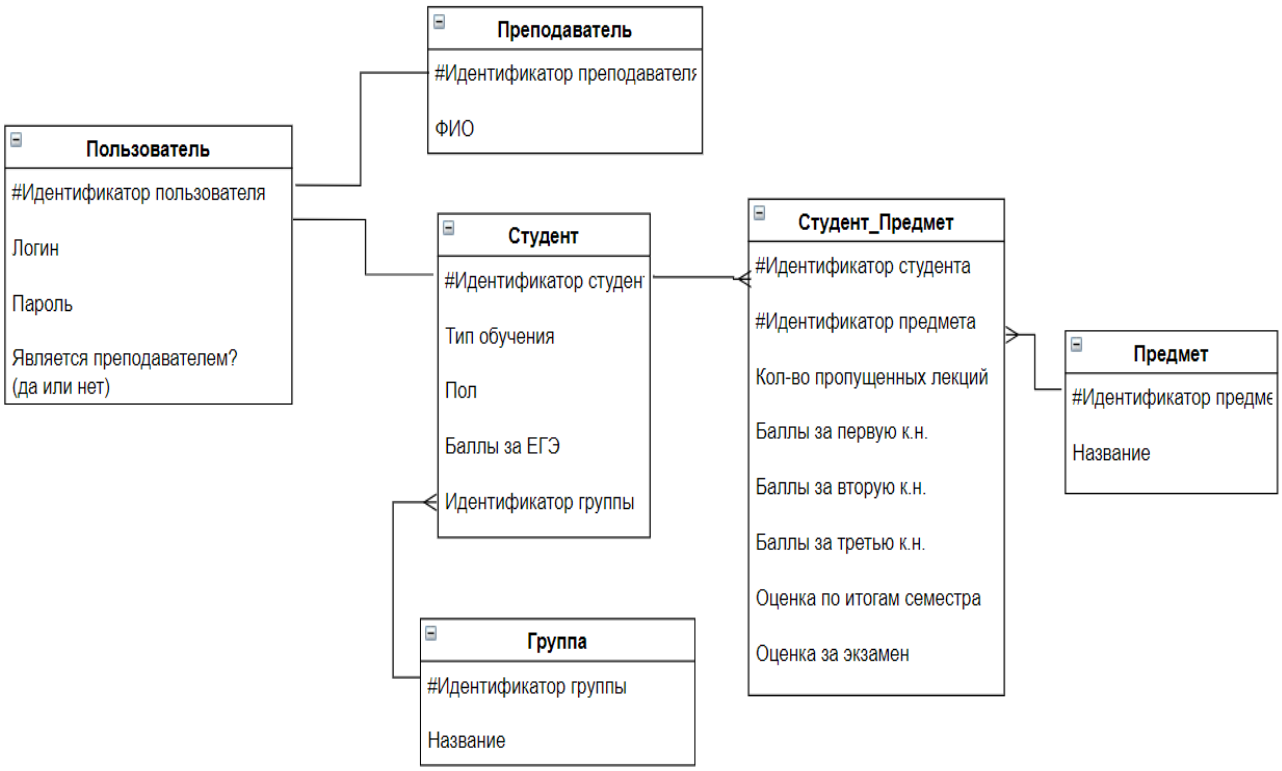

Рис. 3.2. Схема базы данных в нотации ERD

В приведенной, на рисунке 3.2, диаграмме показаны связи между объектами и описаны основные их атрибуты. Пример записи из базы данных выглядит следующим образом (рисунок 3.3):

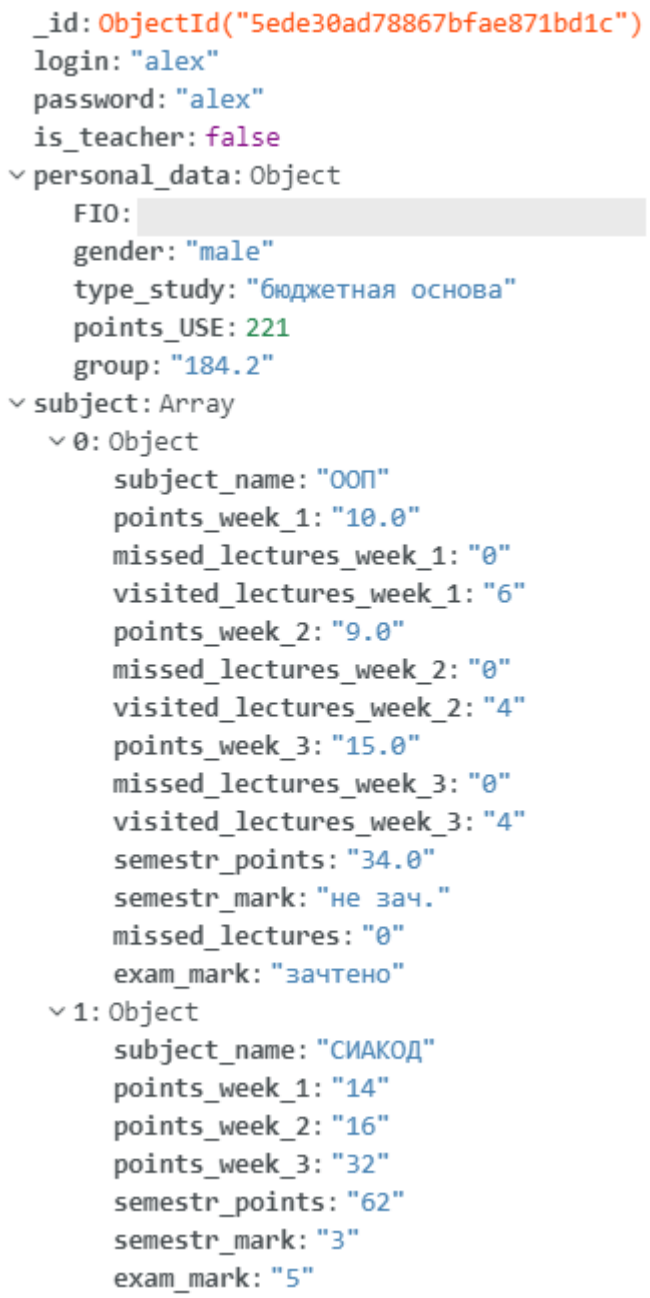

Рис. 3.3. Пример данных, хранящихся в MongoDB

3.3. Используемые технологии для реализации серверной части

В качестве языка программирования для реализации серверной части системы был выбран язык Python.

Python – в настоящее время, один из самых популярных и используемых языков программирования в мире [8]. Он обладает легким синтаксисом, имеет большое число библиотек под разные задачи и является ведущим инструментом в различных областях. Для реализации серверной части Python был выбран как язык, который:

- Является одним из самых используемых языков программирования для создания back-end приложений.
- Имеет большое количество библиотек для работы с данными (анализ, предобработка, визуализация) и для работы с методами интеллектуального анализа данных (машинное обучение, нейронные сети).

Для реализации серверного приложения использовалась библиотека Flask. Это микрофреймворк, предоставляющий базовые возможности по созданию веб-приложений и не использующий никаких дополнительных библиотек.

Вся коммуникация между клиентскими и серверным приложениями заключается в использовании http-протоколов. Для безопасного способа передачи информации между двумя участниками и идентификации клиента используется JSON Web Token (JWT) – JSON объект, закодированный с помощью хеш-функции и содержащий идентификатор пользователя. Данный веб-токен позволяет серверу идентифицировать источник запросов, и при успешной идентификации – разрешает клиенту работу с сервером. Веб-токен генерируется после авторизации пользователя и при дальнейших запросах на сервер его необходимо указывать в заголовке запроса таким образом:

Authorization: <JWT Web Token>.

Основным компонентом разрабатываемой системы является сервис, отвечающий за предобработку и прогнозирование успеваемости студентов. Для работы с данными в языке Python существует большое количество библиотек. Для реализации данной задачи были использованы следующие библиотеки:

- Numpy - низкоуровневая библиотека написанная на C, что дает высокий прирост к производительности вычислений, предназначенная для различных операций над математическими объектами.
- Pandas - высокоуровневая библиотека, построенная над библиотекой Numpy, предоставляющая инструменты для анализа и обработки больших наборов данных.

- Matplotlib и Seaborn - библиотеки для построение различных видов графиков.
- Sklearn - библиотека, предоставляющая широкий выбор алгоритмов машинного обучения.

3.4. Результаты разработки серверной части системы

Было разработано серверное приложение, которое отвечает за:

- Взаимодействие с базой данных, с помощью реализованных классов, которые отвечают за считывание данных из xlsx-файлов, преобразование в формат json и последующего сохранения в базу данных.

  Файл mongoDB.py содержит класс DataBase, предназначенный для работы с базой данных.

  Функции класса:

  - findStudent(fio, group) – поиск студента по переданным ФИО и группе. При успешном выполнении поиска – возвращает информацию о студенте.
  - insertStudent(object) – получает на вход персональную информацию о студенте в JSON формате, проверяет наличие студента в базе, если студент отсутствует – вставляет в базу, иначе обновляет поля студента.
  - insertSubjectInStudent(subject_name, list_columns, object) – в функцию передается название предмета, список полей, которые необходимо вставить и информацию о предмете в JSON формате.
  - authentication(login, password) – функция проверки наличия пользователя в базе данных по логину и паролю.
  - getUserByID(id) – функция получения пользователя по его идентификатору.
  - getUserPersonalInfo(id) – функция получения только персональной информации пользователя по его идентификатору.
  - getUsers(groups) – получение списка студентов по номеру группы.

- updateAuth(fio, group, login, password) – функция обновление логина и пароля студента.

- Взаимодействие с клиентским приложением осуществляется посредством http-запросов, таких как:

  /upload_initial_data [POST] - загрузка файла с данными студентов на сервер, с последующим чтением и записью данных в БД.

  /upload_study_data [POST] - загрузка файла с данными об учебной успеваемости студентов по выбранному предмету.

  /auth [POST] - запрос на авторизацию, ответом которого является веб-токен, который необходим для отправки остальных запросов.

  /fit_model?subject= [GET] - запрос на обучение модели */predict?subject=&group=* [GET] - запрос на прогнозирование результатов обучения студента по выбранному предмету. Если помимо названия предмета передается группа – прогнозируются результаты всей группы (код запроса приведён в приложении 2).

  /get_personal_info [GET] – запрос на получение данных пользователя.

  /get_study_info?subject= [GET] – запрос на получение учебной статистики по выбранному предмету.

  /get_statistics_by_group?subject= [GET] – получение данных по группе пользователя, которая включает в себя: интервалы баллов, количество студентов в них и номер интервала, к которому относится текущий студент.

  /get_user_auth [POST] – запрос на получение логина и пароля пользователя, который идентифицируется по ФИО и номеру группы. Данные передаются в JSON.

  /update_user_auth [PUT] – запрос на обновление логина и пароля пользователя. В теле запроса передаются такие данные, как: ФИО студента, номер группы, измененный логин и пароль.

- Взаимодействие с модулем прогнозирования результатов обучения.

Модуль реализован как класс по паттерну проектирования «Singleton», что позволяет экземпляру класса существовать в единственном виде на протяжении всего приложения. Класс хранит реализованные модели, и, с помощью http-запросов к серверу, вызываются методы данного класса, которые отвечают за взаимодействие с ним.

Реализованные классы для анализа и предобработки данных содержатся в файле machine_learning_service.py.

Классы:

- PreprocessingData – класс отвечает за считывание данных из файлов формата xlsx и за форматирование объектов формата JSON в объекты класса DataFrame из библиотеки pandas.

  Функции класса:

  - addPersonalInfo(file_path) – считывает данные студентов из файла и заносит их в БД.
  - addStudyData(file_path, subject_name) – считывает успеваемость студентов по выбранному предмету из файла и заносит в БД.
  - convertFromJson(json_data, subject_name) – преобразует данные формата JSON в pandas DataFrame.
- Statistics – класс, предназначенный для формирования статистики по данным.

  Функции класса:

  - getStudyData(user_df) – функция на вход получает одного студента в формате DataFrame и возвращает суммарное число полученных баллов и пропущенных лекций.
  - sumValue(df, list_col) – функция отвечает за суммирование значений из выбранных столбцов.
  - splitOnPointsGroup(df, field, count_line) – функция принимает на вход список студентов в формате DataFrame,

поле, по которому необходимо разбить данные на диапазоны, количество диапазонов. Результатом функции являются диапазоны и число студентов в них.

- findInterval(user_value, groups) – относит переданное значение к одному из интервалов. Функция возвращает индекс интервала, к которому относится значение.

- ModelMachineLearning

  Функции класса:

  - dummyCodding(df, subject_name) – заменяет в переданном DataFrame категориальные признаки на вещественные.
  - fillEmptyUSE(df) – функция заполнения пропущенных значений баллов ЕГЭ у студентов.
  - normilizeData(df, subject_name, columns) – функция, отвечающая за нормализацию всех числовых значений.
  - prepareData(df, subject_name, allow_normilize) – предобработка данных с применением трех выше перечисленных методов.
  - prepareModel(df, subject_name) – обучение выбранных моделей машинного обучения.
  - predictLabel(predicted_x, subject_name) – прогнозирование для студентов по выбранному предмету.

Для размещения приложения в сети интернет был создан виртуальный сервер на платформе Digital Ocean с такими характеристиками:

- Операционная система Ubuntu 18.04.3 x64.
- Одноядерный процессор.
- Оперативная память - 1 GB.
- Память на диске – 25 GB.

3.5. Используемые технологии для реализации клиентской части

Многие студенты в современном мире больше всего ценят мобильность и доступность, и приложения, которые всегда будут находиться у них под рукой, являются наиболее актуальными. Именно поэтому было решено реализовать клиентскую часть системы под мобильные платформы ОС Android.

Данное приложение позволит пользователям в любое время получать доступ к актуальной информации об их учебной успеваемости, и также узнавать вероятные исходы их обучения.

Для разработки мобильного приложения использовался быстро развивающийся и набирающий популярность в данной сфере язык программирования - Kotlin.

Kotlin - язык программирования, выпущенный Российской компанией JetBrains в 2016 году, и уже в 2017 году был объявлен Google официальным языком разработки для Android приложений. Он обладает полной совместимостью с Java (компиляция происходит в JVM байткод), что позволяет использовать в одном проекте уже существующие фреймворки и библиотеки, написанные на Java.

Архитектура приложения строится на паттерне проектирования MVP (рисунок 3.4), который позволяет разделить логику работы приложения на разные слои, что облегчает разработку и тестирование. Приложение делится на 3 слоя:

1) Model - для работы с данными.
2) View - для отображения данных и взаимодействия пользователя с интерфейсом.
3) Presenter - слой, отвечающий за внутреннюю логику работы приложения. Является связующим звеном для Model и View.

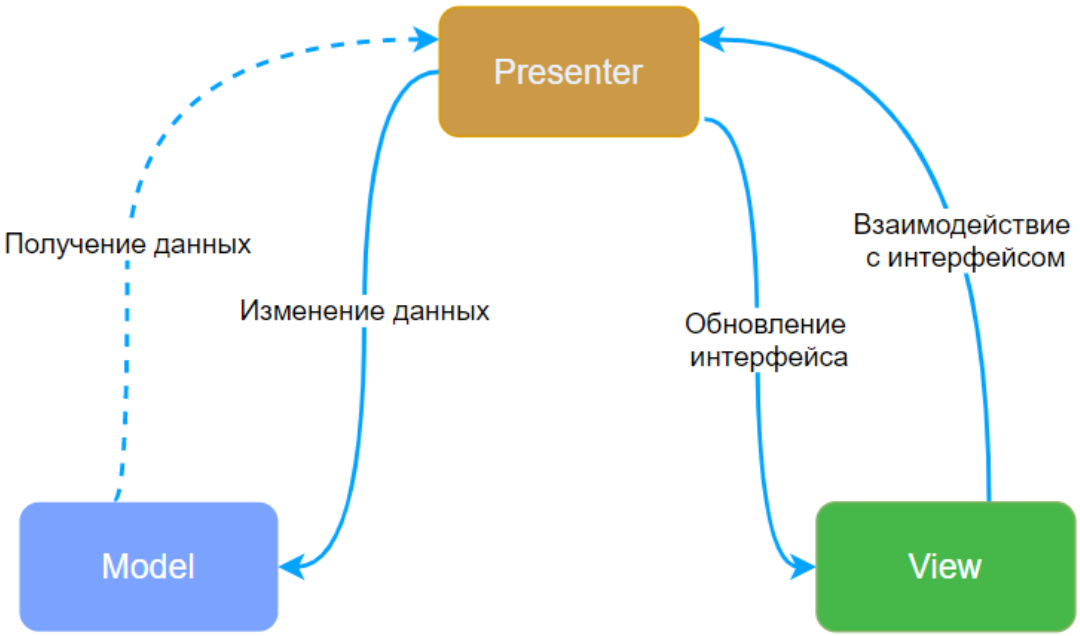

Рис. 3.4. Паттерн проектирования - MVP

Основной функционал мобильного приложения состоит в том, чтобы получить данные с удаленного сервера и отобразить в визуально красивом и понятном виде для пользователя. Загрузка данных должна происходить асинхронно, чтобы пользователь понимал, что приложение не зависло, а работает с сервером (код класса для асинхронных запросов представлен в приложении 3). Для реализации данных задач, могут существенно помочь наиболее используемые библиотеки в мобильной разработке, такие как:

- Retrofit 2 - библиотека для сетевого взаимодействия, с поддержкой REST API.
- Coroutines - библиотека, используемая для реализации параллельного и асинхронного программирования с помощью «корутин» или сопрограмм - легковесных потоков.
- MPAndroidChart - библиотека, предназначенная для построения графиков.

3.6. Результаты разработки клиентской части системы

В качестве клиентской части системы, было реализовано мобильное приложения на платформе Android, написанное на языке Kotlin.

Приложение различает два вида пользователей: студенты и преподаватели и предоставляет им различные функции:

Для студентов приложение позволяет:

- Получить текущую статистику обучения конкретного студента по выбранному предмету.
- Получить прогноз оценок по выбранному предметам.
- Увидеть сравнительные графики успеваемости своей группы.

Для преподавателей функции приложения будут другими, они предоставят возможность:

- Получить прогноз оценок студентов выбранной группы.
- Настроить каждому студенту логин с паролем.
- Посмотреть по каждому студенту подробную статистику обучения.

Для работы с мобильным приложением пользователям необходимо пройти авторизацию. При успешной авторизации, сервер возвращает сгенерированный на основе идентификатора пользователя веб-токен, который позволяет приложению обращаться к защищенным запросам. Также сервер отправляет роль пользователя, благодаря которой приложение открывает соответствующие экраны для разных видов пользователей. Окно авторизации представлено на рисунке 3.5.

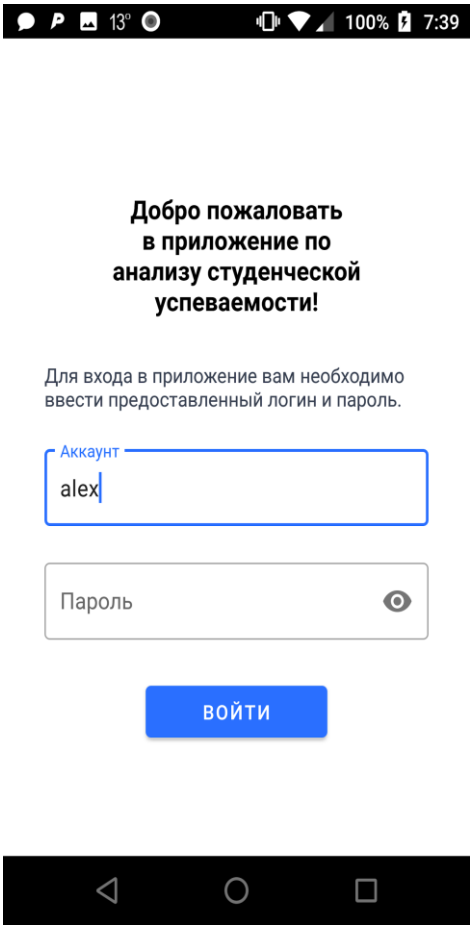

Рис. 3.5. Окно авторизации в мобильном приложении

Если пользователь является студентом, то после успешной авторизации, ему доступно:

- Основное меню, в котором ему предложены предметы на выбор, по которым он может получить текущую успеваемость и прогноз, построенный на её основе. Скриншот приложения приведён на рисунке 3.6.

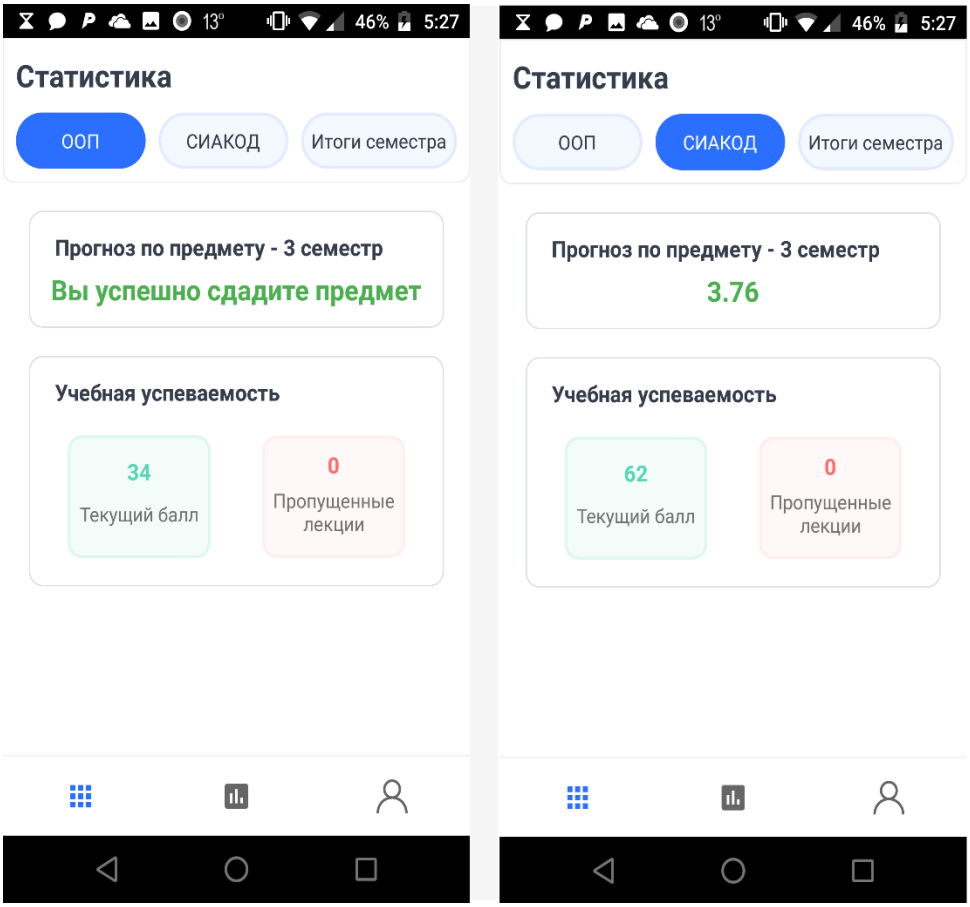

Рис. 3.6. Основное окно приложения

- Меню с графиками, на которых он может сравнить свои учебные результаты с одногруппниками. На рисунке 3.7. приведен пример графика успеваемости, на котором студент может увидеть сколько баллов было получено за контрольную неделю и какие баллы получили его одногруппники.

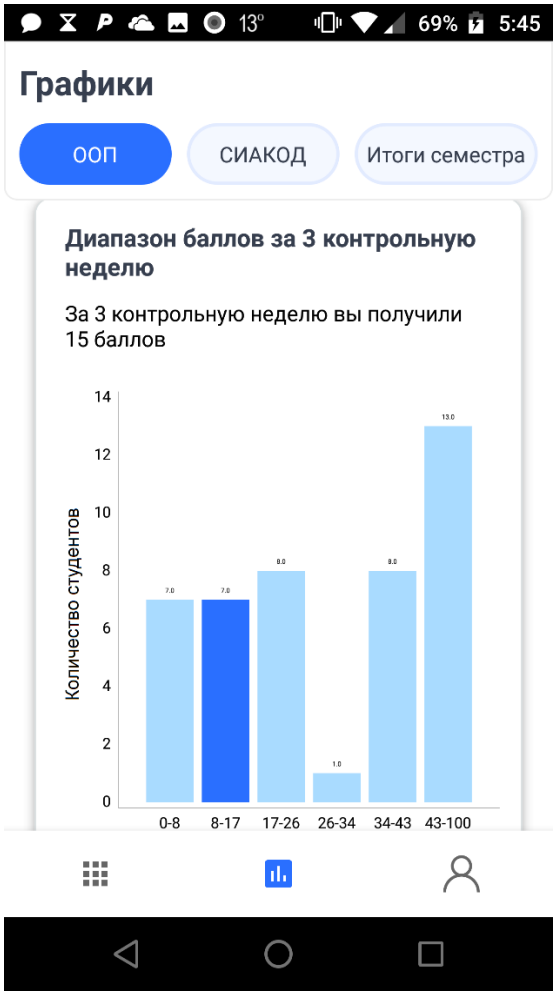

Рис. 3.7. Окно приложения с графиками успеваемости

- Окно с профилем, которое содержит персональные данные авторизованного студента, и в котором, можно прочитать краткую справку по приложению (рисунок 3.7.)

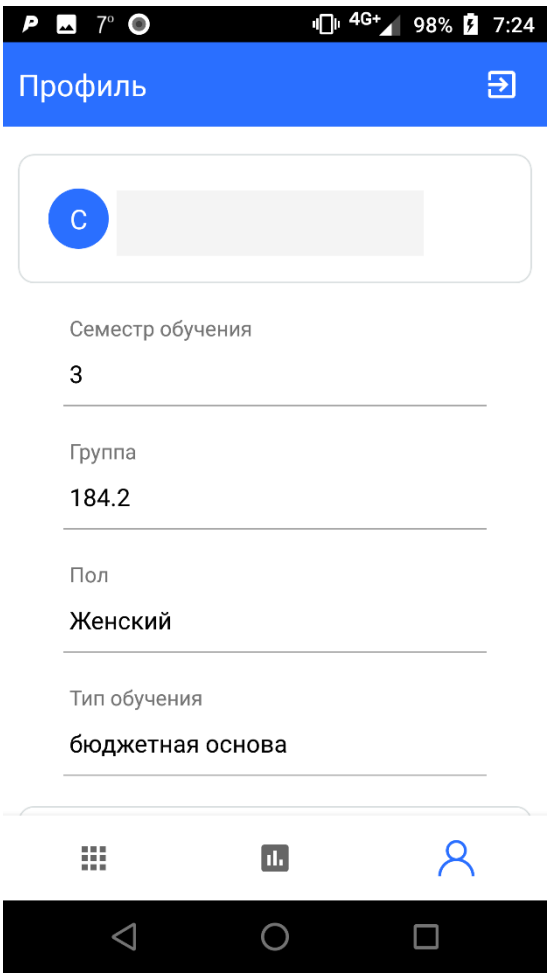

Рис. 3.7. Профиль пользователя

Если пользователь – преподаватель, то при входе в приложение ему будет представлено главное окно, на котором есть возможно выбрать предмет и группу для загрузки результатов по ней (рисунок 3.8.).

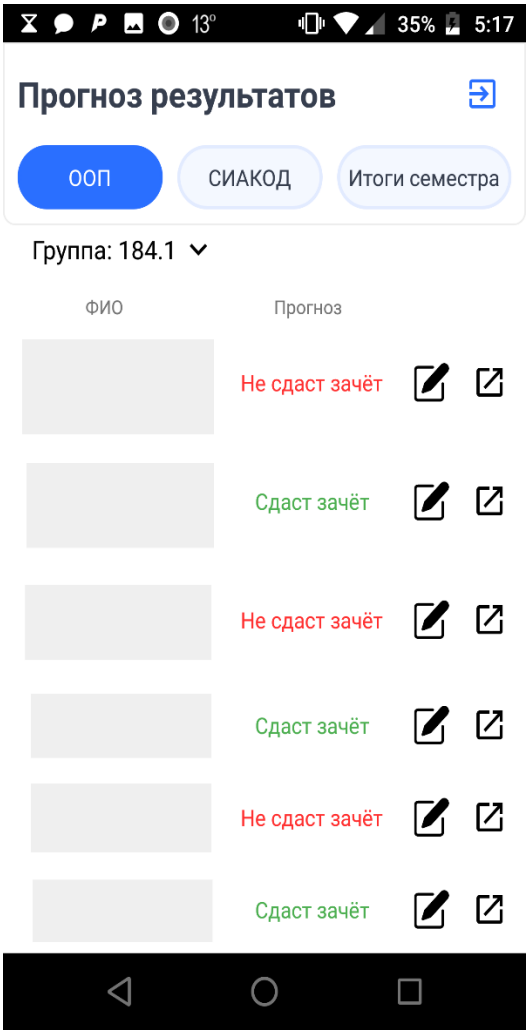

Рис. 3.8. Окно с результатами прогноза зачётов

ЗАКЛЮЧЕНИЕ

В ходе дипломной работы были изучены различные подходы и методы из области машинного обучения и анализа данных. Были собраны данные студентов 2 курса за 3 семестр, проведены анализ и предобработка данных.

Для прогнозирования успеваемости студентов были решены следующие задачи:

- Задача бинарной классификации результата зачёта по предмету «Объектно-ориентированное программирование».
- Задача регрессии для прогнозирования оценки за экзамен по предмету «Структуры и алгоритмы компьютерной обработки данных».
- Задача бинарной классификации по итогам семестра (возможность отчисления).

Для решения задачи бинарной классификации были выбраны следующие методы: KNN, SVM, Logistic Regression, Random Forest. Было проведено сравнение результатов работы выбранных методов с помощью метрик оценивания качества классификации и выбраны модели, показавшие наилучшие результаты. Для решения задачи регрессии был применён метод Linear Regression с модификациями (Lasso и Ridge Regression). Оценка качества регрессионных методов проводилась с помощью таких метрик, как коэффициента детерминации и среднеквадратичной ошибки. Лучшие результаты среди выбранных методов регрессии показали стандартный метод линейной регрессии и его модификация – Ridge Regression.

Полученные результаты исследования позволили реализовать систему для анализа и прогнозирования успеваемости студентов, и при дальнейшем развитии работы можно улучшить точность и качество моделей путем увеличения количества и качества предоставляемых для обучения моделей данных.

Для реализации данной системы была спроектирована база данных для хранения информации о студентах; была разработана серверная часть системы, для взаимодействия с базой данных и клиентским приложением; было

разработано мобильное приложение в качестве клиентской части системы. Разработанное серверное приложение было размещено на платформе Digital Ocean, что позволяет при наличии интернета использовать клиентскую часть системы с мобильных устройств с ОС Android.

В качестве дальнейшего развития системы можно реализовать клиентское приложение на мобильные устройства с операционной системой iOS, увеличить список предметов, для которых строится прогноз и добавить различные графики для наглядного отображения учебных результатов студентов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Ammar Almasri, Erbug Celebi, Rami S. Alkhawaldeh EMT: Ensemble Meta-Based Tree Model for Predicting Student Performance. 2019. 11 с.
2. Bustamante J. College Dropout Rates // educationdata.org: [сайт], 2019. URL: https://educationdata.org/college-dropout-rates/ (дата обращения: 19.03.2020)
3. Canbek G. [и др.]. Binary Classification Performance Measures/Metrics: A comprehensive visualized roadmap to gain new insights. // 2017 International Conference on Computer Science and Engineering (UBMK). 2017.
4. CASE STUDY I: Predictive analytics at Nottingham Trent University // Learning Analytics in Higher Education. 2015.
5. Cornelia G., [и др.]. A Comparative Study: MongoDB vs. MySQL // The 13th International Conference on Engineering of Modern Electric Systems. 2015.
6. Dalson. B., SILVA José. A. What is R2 all about? // Leviathan-Cadernos de Pesquisa Polútica. 2011.
7. Davy Cielen, Arno D. B., Mohamed Ali Introducing Data Science. 320 с.
8. insights.stackoverflow: [сайт]. Developer Survey Results 2019. 2019. URL: https://insights.stackoverflow.com/survey/2019#most-popular-technologies (дата обращения: 18.03.2020)
9. Hao K. What is machine learning? // technologyreview.com [сайт]. 2018. URL: https://www.technologyreview.com/2018/11/17/103781/what-is-machine-learning-we-drew-you-another-flowchart/ (дата обращения: 26.03.2020)
10. Koehrsen W. An Implementation and Explanation of the Random Forest in Python // towardsdatascience.com: [сайт]. 2018. URL: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76 (дата обращения: 27.04.2020)
11. Lubna Mahmoud Abu Zohair Prediction of Student's performance by modelling small dataset size // International Journal of Educational Technology in Higher Education. 2019. С. 16.

12. machinelearning.ru: [сайт]. Коэффициент детерминации. URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%BE%D1%8D%D1%84%D1%84%D0%B8%D1%86%D0%B8%D0%B5%D0%BD%D1%82_%D0%B4%D0%B5%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%B0%D1%86%D0%B8%D0%B8 (дата обращения: 02.06.2020)
13. Megan Squire, Mastering Data Mining with Python – Find patterns hidden in your data. - Published by Packt Publishing Ltd. 2016. 268 с.
14. Nabi J. Machine Learning — Multiclass Classification with Imbalanced Dataset // Towardsdatascience.com: [сайт]. 2018. URL: https://towardsdatascience.com/machine-learning-multiclass-classification-with-imbalanced-data-set-29f6a177c1a (дата обращения: 13.04.2020)
15. Piatetsky G. CRISP-DM, still the top methodology for analytics, data mining, or data science projects // kdnuggets.com: [сайт]. 2014. URL: https://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html (дата обращения: 13.05.2020)
16. Suzanne van den Bosch: Automatic feature generation and selection in predictive analytics solutions. 2017.
17. Wesam S. Bhaya Review of Data Preprocessing Techniques in Data Mining // Journal of Engineering and Applied Sciences. 2017. 12 с.
18. Дунаева К.И., Монанкова Д.Ю. Оценка риска отчисления студентов вузов России: основные причины и последствия. // scienceforum.ru: [сайт]. 2018. URL: https://scienceforum.ru/2018/article/2018008328 (дата обращения: 26.03.2020)
19. Рашка С. Р. Python и машинное обучение / пер. с англ. А. В. Логунова. - М.: ДМК Пресс, 2017. 418 с.
20. Русаков С.В., Русакова О.Л., Посохина К.А. Нейросетевая модель прогнозирования группы риска по успеваемости студентов первого курса // Современные информационные технологии и ИТ-образование. 2018. 815-822 с.

Приложение 1

Методы предобработки данных fillEmptyUSE, normilizeData

```
def fillEmptyUSE(df):
    groups = set()
    for val in df["group"].unique().tolist():
        group =  int(val)
        groups.add(group)

    groups = list(groups)
    for group in groups:
        mean_group = df[df['group'].astype(int) == int(group)]["USE"].mea
n()
        df.loc[(df['group'].astype(int) == int(group)) & (np.isnan(df["US
E"])), "USE"] = int(mean_group)
    return df

def normilizeData(df, columns):
    for col in columns:
        min_val = df[col].min()
        max_val = df[col].max()
        df[col] = df[col].apply(lambda x: (x-min_val)/(max_val-min_val));
    return df
```

Приложение 2

Запрос на прогнозирование оценки по выбранному предмету

```
@app.route("/predict", methods=["GET"])
@jwt_required
def predict():
    subject_name = request.args.get('subject', default=None, type=str)
    if subject_name == None:
        return jsonify({"msg": "Укажите предмет, по которому вы хотите по
лучить прогноз таким образом: /predict?subject=ООП"}), 400
    if subject_name not in app.config["SUBJECTS"]:
        arr = app.config["SUBJECTS"]
        return jsonify({"msg": f"Выбранного вами предмета нет в списке -
{arr}"}), 400
    group = request.args.get('group', default=None, type=str)
    users = None
    db = DataBase()

    if group != None:
        users = db.getUsers(groups=group)
    else:
        id = get_jwt_identity()
        users = db.getUserByID(id)
    df = PreprocessingData().ConvertFromJson(users, subject_name)
    fio_list = df["FIO"].values.tolist()
    group_list = df["group"].values.tolist()
    predicted_x = ml.prepareData(df, subject_name)
    result = ml.predictLabel(predicted_x, subject_name)

    if group != None:
        list_respone = list()
        for ind, val in enumerate(fio_list):
            string = {"fio": val, "group":str(group_list[ind]), "subject_
name": subject_name, "predict": str(round(result[ind], 2))}
            list_respone.append(string)

        return jsonify(list_respone), 200
    else:
        return jsonify({"subject_name": f"{subject_name}", "predict":f"{r
esult[0]}"}), 200
```

Приложение 3

Класс для асинхронного и параллельного выполнения основных запросов интерфейса студента

```
class HomeRepository @Inject constructor(private val retrofit: Retrofit,
private val gson: Gson) {
    lateinit var presenter: HomePresenter

    init {
        App.appComponent.inject(this)
    }

    fun getStudyData(subject_name:String){
        val serv = retrofit.create(ApiService::class.java)
        GlobalScope.launch(Dispatchers.IO) {
            launch {
                try{
                    val result =  async{
                        requestGetStudyInfo(serv, subject_name)
                    }
                    val result2 = async<String> {
                        requestGetPredict(serv, subject_name)
                    }
                    val data = listOf<Any>(result.await(),
result2.await())
                    withContext(Dispatchers.Main){
                        presenter.OnFinished(data)
                    }
                 }
                catch (exception: Exception){
                    Log.e("Exception request", exception.message!!)
                }
            }
        }
    }

    suspend fun requestGetStudyInfo(serv:ApiService, subject_name:
String): StudyInfo{
        val response = serv.GetStudyInfo(subject_name)
        val res = if(response.isSuccessful)
            response.body()!!
        else
            throw Exception("Ошибка при запросе /get_study_info")
        return res
    }

    suspend fun requestGetPredict(serv:ApiService, subject_name: String):
String{
        val response = serv.GetPredict(subject_name)
        val res = if(response.isSuccessful){
            val json = response.body()?.string() ?: ""
```

```
            val objJson = gson.fromJson(json,
JsonElement::class.java).asJsonObject
            objJson["predict"].asString
        }
        else
            throw Exception("Ошибка при запросе /predict")
        return res
    }
}
```