Санкт-Петербургский государственный университет

Жариков Даниил Сергеевич

Выпускная квалификационная работа

Разработка платформы для предиктивного анализа в киберспорте

Уровень образования: бакалавриат

Направление 02.03.03 “Математическое обеспечение и администрирование информационных систем”

Основная образовательная программа СВ.5006.2019 “Математическое обеспечение и администрирование информационных систем”

Научный руководитель:
к.ф.-м.н, доцент кафедры информатики Д. А. Григорьев

Рецензент:
PhD in Computer Science, The University of Florida Александр Семенов

Санкт-Петербург

2023

Saint Petersburg State University

Zharikov Daniil

Bachelor's Thesis

Development of a framework for predictive analysis in esports

Education level: bachelor

Speciality 02.03.03 "Software and Administration of Information Systems"

Programme CB.5006.2019 "Software and Administration of Information Systems"

Scientific supervisor:
PhD in Computer Science, associate professor Dmitriy Grigoriev

Reviewer:
PhD in Computer Science, The University of Florida Alexander Semenov

Оглавление

Введение

Современные технологии приносят много пользы, значительно облегчая человеческую жизнь. Они упрощают коммуникацию, помогают выполнять деловые задачи, хранить, передавать и находить информацию. Все чаще люди используют их с развлекательной целью. Социальные сети, аудиовизуальный контент и мобильные приложения неизменно пользуются популярностью уже не только среди молодежи.

Несколько обособленно развивается индустрия компьютерных игр. Согласно официальным данным, более 40% жителей планеты являются геймерами [1]. Большой спрос порождает новые предложения: лишь за последние десять лет количество видеоигр увеличилось примерно в три раза. Широкая популярность видеоигр породила новый вид соревнований - киберспорт.

Компьютерный спорт изменил представление о видеоиграх: теперь это не просто веселое времяпрепровождение, но и огромное поле для заработка. Игроки формируют команды и соревнуются в международных турнирах за денежные вознаграждения, собирая массы болельщиков. Призовые фонды достигают 40 000 000 $ [2].

Крупнейший киберспортивный турнир «The International» [3] проводится в рамках компьютерной игры «Dota2» [4]. Он является одним из главных событий в игровом сообществе, сравнимым по значимости с чемпионатом мира. Победа на турнире является наивысшим достижением для участников не только в профессиональном плане. Призовой фонд «Dota2» является рекордным, а количество соперников на турнире велико. Зрелищный турнир привлекает миллионы болельщиков из разных стран. «The International» задает тренды в киберспортивной отрасли и популяризует игры.

Как и другие виды спорта, киберспорт активно монетизируется через букмекерские платформы. Люди с азартом предугадывают победителя в намерении заработать посредством ставок. Существуют специальные сервисы, где зрители делятся своими предположениями по поводу результатов различных матчей. Кроме того, в свободном доступе содержится немало информации об уже прошедших играх и их участниках: статистика, предпочтения игроков в выборе персонажей, командные встречи и многое другое [5]. Доступ к подобной информации является важным преимуществом «Dota2» на конкурирующем рынке видеоигр.

Современные методики машинного обучения (machine learning) служат опорой для предсказания будущего результата матча. Компетентные люди

активно пользуются искусственным интеллектом и часто выдвигают правильные заключения, приводящие к значительным выигрышам.

Бурная деятельность вокруг «Dota2» все чаще привлекает внимание научного сообщества. Ученые тщательно изучают киберспорт под разными призмами - с психологической [6], социальной [7] и экономической [8] точки зрения. Киберспорт рассматривается как феномен нашего века [9], требующий научных объяснений.

Данная дипломная работа посвящена разработке платформы для предиктивного анализа в киберспорте.

1. Постановка задачи

Целью работы является создание платформы для предсказания побед команд в киберспортивных состязаниях на основе предматчевых данных, а также для выбора внутриигровых персонажей для подготовки команды к матчу.

Для достижения поставленной цели были сформулированы следующие задачи.

- Произвести обзор научных исследований, посвященных применению различных моделей машинного обучения для прогнозирования исходов соревнований в спортивных и киберспортивных дисциплинах.
- Изучить программные интерфейсы для получения данных из матчей игры “Dota 2”.
- Выбрать архитектурный стиль и создать архитектуру платформы.
- Спроектировать и реализовать интерфейс для подключения предиктивных моделей.
- Провести апробацию платформы.

2. Обзор

2.1. Обзор статей на схожую тематику

Так как киберспорт активно развивается лишь в последние годы, статей по анализу такого сегмента игр не так много, а значит найти необходимую статистику и похожие методы не просто. Для того, чтобы понимать, как осуществляется сбор данных, построение моделей для дальнейшего анализа, необходимо было ознакомиться со статьями на подобную тематику.

2.1.1. Анализ статей по футболу

Футбол является самым популярным видом спорта на настоящий день [10]. Победа команды в матче зависит как от личных навыков отдельных игроков, так и от их командного взаимодействия.

В связи с его популярностью было написано множество статей как по анализу матчей [11] и предсказанию побед [12], так и по внутриигровому взаимодействию игроков [13].

Исследования о футболе можно классифицировать по следующим идеям анализа.

- анализ командной составляющей и ее влияние на победу в матче;
- анализ личных навыков игроков и их влияние на победу в матче;
- подготовка футболистов и тренеров к предстоящим матчам, анализ стратегий команд.

Начнем рассмотрение со статей, описывающих и рассматривающих важность командного взаимодействия.

R. Ievoli и L. Palazzo [14] показывают зависимость удержания мяча, количества передач и взаимодействия игроков между друг другом с победой команды в футбольном матче. Сети передачи и их структурные особенности могут использоваться для оценки стиля игры с точки зрения поведения передачи, анализа и количественной оценки взаимодействий между игроками. В [15] команда рассматривается как сложная сеть, узлы которой (игроки) взаимодействуют с целью преодоления сети противника. Авторы статьи смогли определить те сетевые показатели, которые увеличивают вероятность забить, получить гол. J. M. Buldú и J. Busquets

[16] строят анализ пассов, основываясь на направленности передачи, взвешенности, пространственно-временном взаимодействии игроков.

T. Kharrat, I. G.McHale и др. [17] описывают методологию "плюсов и минусов", которая напрямую измеряет вклад игрока в успех команды, измеряемый разницей целевой метрики (например, голов) во время его присутствия на поле. Авторы используют рейтинги "плюсов и минусов" чтобы определить, кто являются лучшими игроками в европейском футболе, и продемонстрировать, как рейтинги игроков меняются с течением времени. В [18] используется уникальный набор данных для определения ключевых членов футбольной команды. Методология использует статистическую модель для определения сложности передачи паса от одного игрока к другому и объединяет эту информацию с результатами сетевого анализа, чтобы определить, какие игроки являются ключевыми для каждой команды английской премьер-лиги в сезоне 2012–2013 годов.

Другие исследования направлены на нахождение действенных, но неизвестных тренеров, которые дают большой прирост команде. Предоставляется анализ сетей, построенных на взаимодействии тренеров и футбольных команд из НФЛ, который в какой-то мере показывает, какие команды выйдут в плей-офф [19]. B. Gonçalves и D. Coutinho [20] показывают, как сети передачи и переменные позиционирования могут быть связаны с исходом матча в футболе молодежных элитных ассоциаций. Сетевой подход в рамках переменных, определяемых позиционированием, был рассчитан для определения вклада отдельных игроков в общий результат поведения команды во время эмулируемого матча. Результаты авторов показали, что высокая степень взаимосвязанности внутри команды были связаны с лучшими результатами.

Результаты исследований, описанных авторами в вышеупомянутых статьях, свидетельствуют о том, что анализ математических моделей, построенных на внутриигровых данных, позволяет извлекать дополнительную статистическую информацию о спортивных состязаниях. Эта информация может быть структурирована и использована, например, для предсказания победы одной из команд.

2.1.2. Анализ статей по Dota 2

Ученые занимались вопросом предсказания победы одной из команд по определенным предматчевым данным. Основной идеей было использование различных моделей машинного обучения для получения наилучшего результата. Факторов, способствующих победе одной из сторон, бессчетное множество, исследователи старались выделить важность определенных факторов над другими, показать реальные модели, дающие определенные результаты. Подходов к анализу было описано множество, анализ таких статей представлен ниже.

Исследователи K. Conly и D. Perry [21] разработали логистическую регрессию и обучили ее на наборе данных, содержащем почти 57 тысяч записей о предматчевой статистике. Это позволило достичь точности предсказания победы одной из команд на уровне около 70%. Исследование также подчеркнуло важность учета информации из стадии драфта при предсказании победы одной из команд.

Другие исследователи, A. Agarwala и M. Pearce [22], включили в модель логистической регрессии взаимодействия между героями, что позволило определить роль каждого героя и смоделировать их взаимодействия.

Исследователи G. A. Aryanata и др. [23] провели исследование, в котором использовали метод AHP (Analytical Hierarchy Process) для предсказания победы команд в киберспортивных матчах. В качестве метрик были использованы последние 3 матча между командами, 10 последних матчей каждой команды, средний GPM, XPM и средний рейтинг игроков (MMR). Данный метод показал хорошую эффективность: 75%.

Исследователи N. Kinkade и K. K. Lim [24] провели сравнение результатов теста различных послематчевых метрик, таких как GPM, XPM, KPM каждой команды и каждого игрока. В их исследовании были протестированы две модели: логистическая регрессия и random forest classifier. Обе модели показали хороший результат: более 99,5%, что говорит об эффективности предсказания матча на основе послематчевых данных.

Исследователи K. Akhmedov и A. H. Phan [25] занимались предсказанием победы, основываясь на онлайн (real-time) информации, то есть в каждой временной точке игры. Для этого они протестировали 3 модели: Linear

Regression (LR), Neural Networks (NN) и тип рекуррентной нейронной сети Long Short-Term Memory (LSTM). Полученная точность зависела от множества предиктивных параметров, где в худшем случае линейная регрессия показала 69%, а среднее – 82%. Модели глубокого обучения продемонстрировали результаты в 88% (Neural Networks) и 93% (LSTM).

Информация, полученная из вышеупомянутых статей, свидетельствует о том, что киберспортивные события могут быть предсказаны с определенной вероятностью при использовании различных моделей машинного обучения.

2.2. Обзор программных интерфейсов для получения данных о матчах игры “Dota 2”.

OpenDota API [26] и Steam Web API [27] являются программными интерфейсами, которые предоставляют доступ к информации о Dota 2. Оба они предоставляют данные о матчах, игроках, героях и предметах. Однако между этими интерфейсами есть следующие различия.

1. OpenDota API более гибкий и позволяет получать данные в более удобном формате, таком как JSON или CSV. В то время как Steam Web API (Dota 2) предоставляет данные только в формате XML.
2. OpenDota API имеет больше функций, таких как поиск игроков по имени или ID, анализ матчей и предоставление статистики по героям. В то время как Steam Web API (Dota 2) предоставляет только базовую информацию о матчах и игроках.
3. OpenDota API более надежный и стабильный, чем Steam Web API. Это связано с тем, что OpenDota API использует кэширование данных, что позволяет уменьшить нагрузку на серверы Dota 2.
4. Steam Web API требует авторизацию через Steam, что может быть неудобно для пользователей, которые не хотят связывать свои данные с аккаунтом Steam. В то время как OpenDota API не требует авторизации.

В целом, оба программных интерфейса предоставляют полезную информацию для фанатов Dota 2. Однако, если нужны более детальные данные и больше функций, то предлагается использовать OpenDota API. Если же нужна только базовая информация о матчах и игроках, то можно использовать Steam Web API.

В текущем проекте было решено использовать OpenDota API, так как детальность матчей критически важна для обучения модели, прогнозирующей победу команды в матче.

3. Архитектура

В данном разделе рассмотрено сравнение архитектурных стилей и выбор одного из них, а также разделение ответственностей внутри платформы и описание каждого модуля платформы.

3.1. Сравнение микросервисной и монолитной архитектур

Существует множество подходов к созданию программного обеспечения. В зависимости от подхода различаются многие факторы, такие как сложность проектирования, стоимость разработки, скорость ответа и многие другие. У микросервисной и монолитной архитектур есть как преимущества, так и недостатки перед друг другом, с которыми необходимо разобраться.

3.1.1 Монолитная архитектура

Данный архитектурный стиль подразумевает создание системы как единого целого, где внутри одной кодовой базы содержится вся бизнес-логика. Чтобы внести изменения в приложения, построенные на такой архитектуре, необходимо обновить весь стек внутри всей кодовой базы и в дальнейшем развернуть обновленную версию всего серверного (в текущем случае) приложения. Это делает обновления ограничительными и отнимающими много времени.

Монолитная архитектура имеет следующие преимущества:

- простота развертывания,
- простота разработки,
- производительность,
- простое тестирование,
- простое отлавливание ошибок.

Недостатками монолитной архитектуры является следующее:

- уменьшение скорости разработки при увеличении проекта,
- расширяемость,

- отказоустойчивость,
- сложность внедрения новых технологий,
- недостаток гибкости.

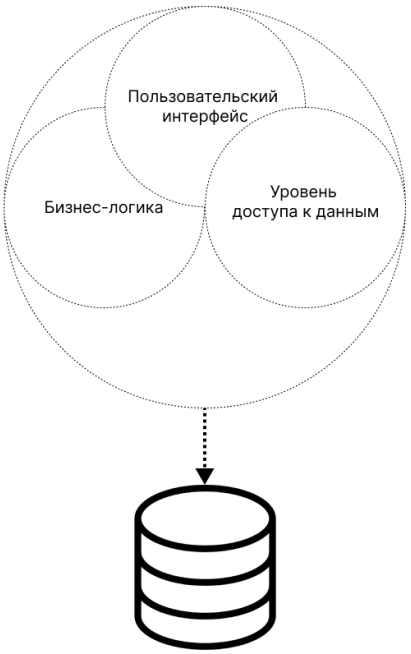

Рис. 1: Монолитная архитектура

Монолитные приложения могут быть удобны при разработке небольших проектов из-за простоты управления кодом, развертыванием приложения на сервере, но неудобны при разрастании проектов по мере их жизни.

3.1.2 Микросервисная архитектура

Данный архитектурный стиль (часто его называют просто микросервисы)подразумевает, что вся бизнес-логика и второстепенные задачи разбиваются на небольшие службы, которые связаны между собой и активно друг друга вызывают. Каждый такой сервис имеет собственную кодовую базу и собственную базу для хранения данных. Микросервисная архитектура позволяет более эффективно управлять сложностью путем разделения приложения на более мелкие и независимые компоненты (микросервисы), каждый из которых выполняет свою определенную функцию. Это упрощает разработку, тестирование и поддержку приложения, поскольку каждый микросервис может быть разработан и поддерживаться отдельно от других..

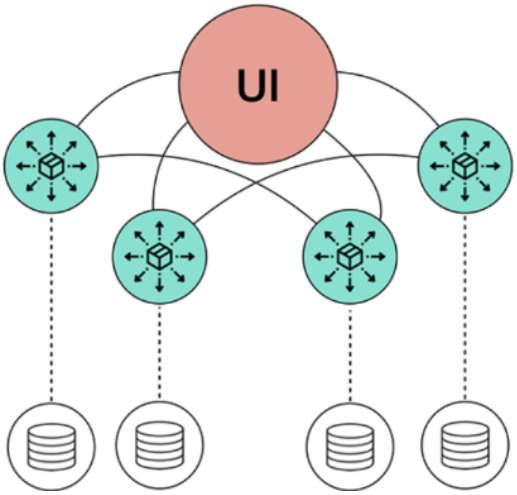

Рис. 2: Микросервисная архитектура

Преимуществами микросервисной архитектуры являются следующие особенности:

- гибкость разработки,
- гибкое масштабирование,
- непрерывное развертывание,
- высокая ремонтопригодность и возможность тестирования,
- независимое развертывание,
- гибкость в используемых технологиях,
- низкая отказоустойчивость.

Недостатками микросервисной архитектуры является следующее:

- разрастание разработки,
- высокие затраты на инфраструктуру приложения,
- дополнительные организационные расходы,
- сложность отлавливания ошибок,
- недостаток стандартизации.

В рамках данной работы был выбран микросервисный архитектурный стиль, в первую очередь, из-за его гибкости и масштабируемости. Микросервисы могут быть разработаны и развернуты независимо друг от друга, что позволяет быстро и легко вносить изменения в систему. Кроме того, микросервисы могут быть написаны на разных языках программирования и использовать разные технологии, что позволяет выбирать наиболее подходящие инструменты для каждого сервиса.

3.2. Построение архитектуры платформы

Разработка архитектуры является важным этапом в любом проекте, особенно в проектах с микросервисной архитектурой. Это связано с тем, что в таких проектах используется множество микросервисов, которые взаимодействуют друг с другом. Каждый микросервис выполняет свою функцию и имеет свою собственную базу данных.

Правильно разработанная архитектура позволяет упростить разработку, тестирование и развертывание микросервисов. Она также обеспечивает масштабируемость и устойчивость системы к сбоям. Кроме того, хорошо спроектированная архитектура позволяет легко добавлять новые микросервисы и изменять существующие без нарушения работы системы в целом.

Распределение обязанностей между сервисами в системе представляет собой четкую и оптимизированную структуру, где каждый сервис выполняет свою уникальную функцию.

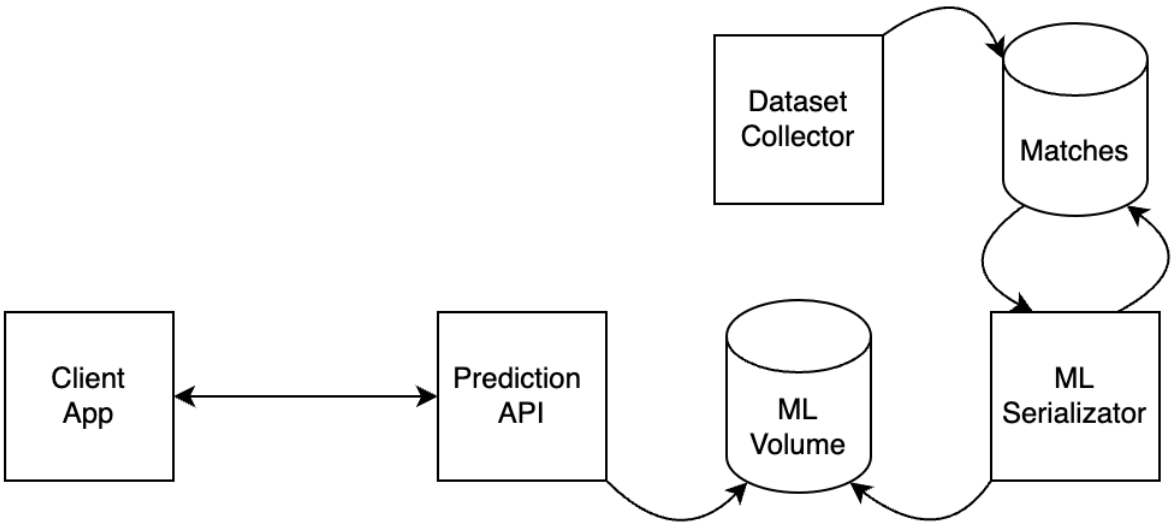

Рис. 3: Диаграмма распределения сервисов

3.2.1. Dataset Collector

Этот сервис является сборщиком данных о матчах Dota 2 и отвечает за регулярное получение информации о прошедших играх из открытых программных интерфейсов. Основной задачей данного сервиса является сбор и хранение развернутой информации о матчах в базе данных для последующего использования другими сервисами.

3.2.1.1. Выбор средств разработки

Для реализации сервиса по регулярному сбору данных о прошедших матчах по игре “Dota 2” было решено использовать фреймворк ASP.NET [28]. Данный фреймворк предназначен для создания веб-приложений, который предоставляет разработчикам множество инструментов и библиотек для упрощения процесса разработки. Платформа имеет встроенную поддержку кэширования, что позволяет уменьшить количество запросов к базе данных, а это крайне важно.

3.2.1.2. Выбор ORM и хранилища данных

Хранение и доступ к данным является одним из ключевых аспектов при разработке веб-приложения на фреймворке ASP.NET. ORM (Object-Relational Mapping) позволяет связать объекты приложения с таблицами в базе данных, что упрощает работу с данными и ускоряет процесс разработки. В данном случае, рекомендуется использовать Entity Framework[29] – это ORM, разработанная Microsoft, которая интегрирована в ASP.NET и обеспечивает удобный и эффективный доступ к данным.

Однако, помимо ORM, необходимо выбрать и хранилище данных. В данном случае, рекомендуется использовать PostgreSQL - это мощная и надежная реляционная база данных с открытым исходным кодом. PostgreSQL обладает широкими возможностями по работе с данными, поддерживает множество типов данных и имеет высокую производительность.

Таким образом, выбор Entity Framework и PostgreSQL [30] является оптимальным решением для разработки веб-приложения на фреймворке ASP.NET. Эти технологии обеспечивают удобный и эффективный доступ к данным, а также гарантируют надежность и целостность данных.

3.2.1.3. Выбор библиотеки планирования задач

Coravel[31] и Quartz.NET[32] – это две популярные библиотеки для планирования задач в .NET. Обе библиотеки обладают широкими возможностями и хорошей документацией, но имеют свои особенности.

Quartz.NET – это мощный и гибкий планировщик задач, который позволяет создавать сложные расписания и управлять задачами из кода. Он поддерживает множество типов триггеров и позволяет настраивать

множество параметров. Однако, Quartz.NET имеет достаточно высокий порог входа и может быть сложен в использовании для начинающих разработчиков.

Coravel – это легковесная и простая в использовании библиотека для планирования задач. Она предоставляет удобный API для создания и управления задачами, а также поддерживает множество типов триггеров. Coravel также обладает широкими возможностями и хорошей документацией, но при этом имеет более простой и интуитивно понятный интерфейс.

Исходя из вышесказанного, было решено использоваться Coravel, как планировщик задач. Он является более удобным и легковесным в использовании, что позволяет сократить время на разработку и упростить поддержку кода в будущем. Кроме того, Coravel имеет более простой и интуитивно понятный интерфейс, что делает его более доступным для начинающих разработчиков.

3.2.2. Prediction API

Этот веб-сервис обеспечивает взаимодействие между клиентским приложением и десериализованной моделью машинного обучения, которую предоставляет ML Serializer. Данный сервис реализован в виде программного интерфейса (API), который позволяет осуществлять HTTP (POST) запросы к модели машинного обучения. Он предоставляет либо команду-победителя, либо лучшего к выбору персонажа. Одним из главных преимуществ данного подхода является возможность получения ответа на запрос в любой момент времени, независимо от того, происходит ли обучение модели в данный момент или нет. Таким образом, пользователи могут быть уверены в том, что получат необходимую информацию в максимально короткие сроки.

3.2.2.1. Выбор средств реализации

При выборе средств реализации для программного интерфейса на Python, необходимо учитывать несколько факторов. Важными критериями являются производительность, масштабируемость, удобство использования и поддержка сообществом.

В данном случае, рекомендуется использовать Fast API[33]. Fast API – это легковесный и простой в использовании фреймворк для создания

программных интерфейсов на языке программирования Python. Он обладает высокой производительностью и масштабируемостью, что позволяет обрабатывать большие объемы запросов и обеспечивать быстрый доступ к результатам, а также Fast API имеет широкую поддержку сообщества и активно развивается, что обеспечивает надежность и стабильность фреймворка в долгосрочной перспективе.

3.2.2.2. Десериализация модели машинного обучения

Этот процесс занимается восстановлением сохраненной модели из файла или другого источника данных. При десериализации модели необходимо учитывать формат, в котором она была сохранена, а также выбрать подходящую библиотеку для десериализации.

В данном случае, рекомендуется использовать библиотеку pickle [34] для сериализации и десериализации моделей машинного обучения. Pickle - это стандартная библиотека Python, которая позволяет сохранять и загружать объекты Python в бинарном формате. Она обладает высокой производительностью и простотой в использовании, что делает ее удобной для сериализации и десериализации моделей машинного обучения.

3.2.3. ML Serializer

Данная библиотека предназначена для обработки данных, полученных с помощью Dataset Collector. Данные, собранные сборщиком данных, используются для обучения модели машинного обучения, которая впоследствии сериализуется и сохраняется в специально выделенном месте на сервере. Сериализация модели позволяет сохранить ее состояние и использовать в дальнейшем без необходимости повторного обучения. Такой подход позволяет эффективно использовать собранные данные и повышает качество обучения модели.

3.2.4. Клиентское приложение

Одной из задач клиентского приложения является сбор информации от пользователей о выбранных героях и запрещенных героях. Собранные данные будут преобразованы в формат JSON и отправлены на программный интерфейс Prediction API.

Ответ, полученный от Prediction API, будет визуализирован в пользовательском интерфейсе приложения, что позволит пользователям

быстро и удобно получить необходимую информацию о предсказании победы в игре.

3.2.4.1. Выбор средств разработки

Для разработки клиентского приложения был выбран фреймворк React[35] по следующим причинам.

React обладает высокой производительностью и быстродействием благодаря виртуальной системе DOM [36] и оптимизации рендеринга.

React имеет большое сообщество разработчиков и обширную документацию, что делает его легко доступным и удобным для изучения. Это позволяет быстро решать проблемы и находить ответы на вопросы в процессе разработки, а это крайне важно, так как у автора не было опыта разработки клиентских веб-приложений.

3.2.4.2. Выбор библиотеки для отправки HTTP-запросов

При выборе HTTP-клиента для приложения, написанного на React [35], , необходимо учитывать несколько факторов, таких как производительность и удобство использования. Для отправки запросов на Prediction API, было решено использовать библиотеку Fetch [37].

Библиотека Fetch API встроена в большинство браузеров для отправки HTTP-запросов. Он имеет простой интерфейс и поддерживает промисы, что обеспечивает удобство использования.

3.3. Основной сценарий использования клиентского приложения

Для получения от пользователей информации о выборе героев в матче Dota 2 веб-приложение на React [35] использует специальные поля ввода, где пользователи могут ввести список выбранных героев за тьму и за свет.

Приложение использует компонент формы, содержащий текстовые поля для ввода списка героев и кнопку "Predict". Пользователь может ввести список героев в поле ввода и нажать на кнопку "Predict", чтобы передать выбранные данные на API.

При вводе списка героев пользователь может использовать различные форматы, например, разделять героев запятыми или переносами строк.

Приложение обрабатывает введенные данные и преобразовывает их в формат JSON, который отправляется на API с помощью POST-запроса.

Рис. 4: Пользовательский интерфейс платформы.

После того, как пользователь ввел список героев и нажал на кнопку "Predict", приложение отправляет данные на программный интерфейс с помощью POST-запроса, содержащего JSON-объект с выбранными героями. Для этого была использована библиотека Fetch [37]. После отправки данных на программный интерфейс “Prediction API”, приложение получает ответ с предсказанием победы и отображает его пользователю.

3.4. Интерфейс для подключения предиктивных моделей

Для того, чтобы подключить отличную от используемой в платформе модель машинного обучения, необходимо клонировать репозиторий платформы [38], необходимо убедиться, что модель соответствует нижеописанным требованиям, а также соответствует следующему интерфейсу, а именно реализует следующие два метода.

- *void model.fit(X, y),* где X – это вектор признаков, а y – это метки классов (true или false).
- *double model.predict_proba(X),* где X – это входные данные, а результат – это выходные данные модели.

3.4.1. Обучение модели

Чтобы добавить в платформу модель машинного обучения, она должна быть заранее обучена на данных последнего большого глобального обновления. Получить необходимые данные можно по следующей конечной точке: /download.

Необходимо учесть, что модель будет периодически дообучаться новыми данными: статистикой о прошедших матчах за прошедшие сутки.

3.4.2. Сериализация

Модель машинного обучения должна быть сериализована с использованием библиотеки pickle и перед первым запуском платформы должна заменить используемую в платформе модель в следующей папке “diploma_spbu/src/MLModel/app/model”.

3.4.3. Входные данные

Модель должна иметь возможность принимать на вход данные в формате, используемом в платформе. Входные данные: целочисленный одномерный массив из 10 элементов, где первые 5 являются индексами героев сил света, а остальные индексами героев сил тьмы.

Пример входных данных: [1, 2, 3, 4, 5, 6, 7, 8, 9, 11], где [1, 2, 3, 4, 5] – список героев сил света, а [6, 7, 8, 9, 11] – список героев сил тьмы.

3.4.4. Выходные данные

Модель должна возвращать на выходе прогноз победы сил света или силы тьмы с коэффициентом от 0 до 1, где 0 – безусловная победа сил тьмы, а 1 – победа сил света. Вывод должен осуществляться по следующему запросу: model.predict_proba([radiantHeroes + direHeroes]).

3.4.5. Пример внешней предиктивной модели

Ниже приведен пример предиктивной модели, которая может заменить используемую в платформе модель машинного обучения.

```
import pickle

from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

model.fit(train_x, train_y)

probability = model.predict_proba(X)

pickle.dump(model, open('model-0.1.0.pkl', 'wb'))
```

Листинг 1: Пример предиктивной модели.

4. Особенности реализации

Разработанная платформа для предиктивного анализа была создана с применением микросервисной архитектуры, которая предполагает разделение платформы на отдельные сервисы. Реализация платформы включает в себя не только создание отдельных сервисов, но и настройку их взаимодействия для достижения наилучшей производительности и эффективности.

4.1. Сбор данных и обновление модели машинного обучения

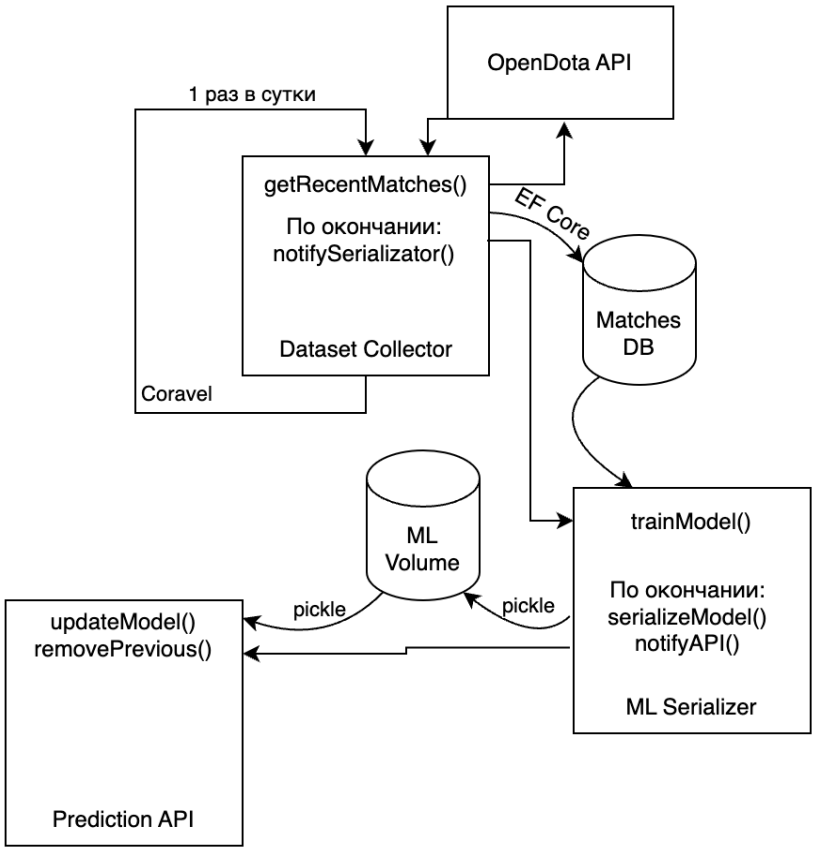

Рис. 5: Процесс получения данных и обновления модели

Для регулярного сбора данных о прошедших матчах и обновления конечной модели необходимо периодически инициализировать сбор данных в Dataset Collector. Запускается процесс сбора новой информации, включая запросы к OpenDota API [13], преобразование данных и сохранение их в базу данных. По завершении сбора данных, Dataset Collector отправляет запрос на ML Serializer, уведомляя сервис о том, что новые данные были собраны.

Затем ML Serializer получает матчи из базы данных, дообучает модель и сохраняет ее в специально выделенное место, общее с Prediction API. После этого ML Serializer отправляет запрос на Prediction API, сообщая, что модель была дообучена и выгружена. Prediction API обновляет текущую десериализованную модель и удаляет предыдущую из хранилища.

4.2. Обработка запроса пользователя

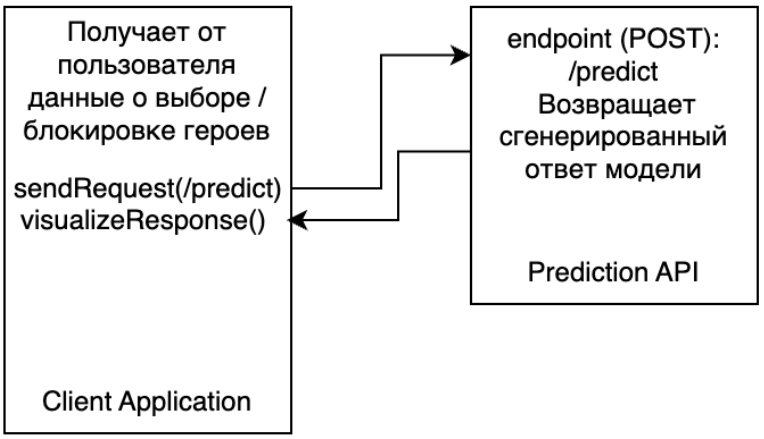

Рис. 6: Процесс обработки запроса пользователя

Клиентское приложение получает списки выбранных и заблокированных героев за свет и тьму в игре Dota 2, осуществляет обработку данных, полученных от пользователей, и формирует JSON-сообщение, которое отправляется на сервис Prediction API. После получения сообщения от клиентского приложения, Prediction API формирует запрос к модели машинного обучения, который позволяет получить ответ и вернуть его обратно в клиентское приложение.

Ответ модели машинного обучения получается на стороне клиента и визуализируется в удобном формате для пользователя.

4.3. Публикация платформы

4.3.1. Контейнеризация сервисов

Контейнеризация микросервисов – это процесс упаковки микросервисов в контейнеры, которые могут быть запущены в любой среде без необходимости установки дополнительных зависимостей. Контейнеры

обеспечивают изоляцию микросервисов и позволяют разработчикам легко переносить их между различными средами.

Одним из наиболее популярных инструментов для контейнеризации микросервисов является Docker[39]. Docker позволяет создавать, управлять и запускать контейнеры, которые могут быть использованы для развертывания микросервисов в любой среде. Контейнеры Docker обеспечивают высокую производительность и масштабируемость, что делает их идеальным выбором для разработки и развертывания микросервисов.

Для упрощения развертывания всей системы было принято решение упаковать все сервисы в соответствующие Docker-контейнеры. Такой подход позволяет легко переносить сервисы между различными средами и обеспечивает высокую производительность и масштабируемость приложения. Контейнеризация сервисов также позволяет быстро масштабировать приложение, добавляя или удаляя контейнеры в зависимости от нагрузки. Это обеспечивает высокую доступность и надежность всей системы.

4.3.2. Развертывание сервисов с помощью Docker Compose

Для настройки непрерывной развертки веб-сервера на Linux системе, было принято решение использовать Docker-Compose[40] для развертывания всех необходимых сервисов.

Все сервисы были упакованы в соответствующие Docker-контейнеры, которые могут быть запущены и управляемы с помощью Docker-Compose. Docker-Compose позволяет определить и настроить все сервисы в одном файле, что упрощает процесс развертывания и управления всей системой.

В файле docker-compose.yml были определены все необходимые сервисы, включая сервис для автоматического парсинга данных матчей, сервис для обучения модели машинного обучения, сервис для предсказания победы и генерации предложений при выборе героев, а также клиентское веб-приложение.

4.3.3. Настройка непрерывного развертывания

Для развертывания микросервисного приложения был выбран Linux сервер на Debian [41]. Это обусловлено тем, что Debian является одним из

наиболее популярных дистрибутивов Linux, который обладает высокой степенью стабильности и безопасности. Более того, Debian имеет обширную базу пакетов, что позволяет легко устанавливать и настраивать необходимые компоненты.

Процесс настройки непрерывного развертывания микросервисного приложения с помощью GitHub Actions[42] состоит из следующих этапов.

1. Подключение к серверу по логину и паролю.
2. Добавление новых пользователей: администратор и github.
3. Настройка подключения к серверу по SSH-ключам.
4. Отключение возможности подключения по логину и паролю. Это позволяет уменьшить риск несанкционированного доступа к серверу, так как SSH-ключи обладают более высоким уровнем безопасности по сравнению с логином и паролем.
5. Авторизация под аккаунтом github и клонирование проекта из репозитория.
6. Установка Docker [38] на сервер.
7. Открытие внешних портов для доступа к сервисам.
8. Запуск склонированного проекта с помощью docker-compose.
9. Добавление SSH-ключа в Github-secret.
10. Настройка Github Actions [41] для выполнения команд развертывания при коммите в основную ветку (main).
11. Тестирование непрерывного развертывания: коммит дополнительного функционала в основную ветку. Проверка на успешное добавление функционала в уже развернутый проект.

Таким образом, процесс настройки непрерывного развертывания микросервисного приложения с помощью GitHub Actions и docker-compose включает в себя ряд шагов, начиная от подключения к серверу и заканчивая тестированием непрерывного развертывания. Для обеспечения безопасности сервера были приняты меры, такие как отключение возможности подключения по логину и паролю и настройка подключения по SSH-ключам. Также были выполнены необходимые настройки для подключения к аккаунтам github и запуска проекта с помощью docker-compose. Настройка Github Actions позволяет автоматизировать процесс развертывания приложения при коммите в основную ветку. Тестирование непрерывного развертывания позволяет убедиться в корректности работы процесса и успешном добавлении новой

функциональности в уже развернутое приложение. Все эти шаги позволяют обеспечить быстрое и безопасное развертывание микросервисного приложения с помощью GitHub Actions и docker-compose.

4.4. Модель машинного обучения

Для предсказания результатов матчей в платформе используется модель машинного обучения, которая была обучена на данных, содержащих информацию о предматчевой статистике в игре, такой как состав команд и их история взаимодействия. Обучение модели проходило на 150 тысячах матчей, а для анализа данных использовалась GNN [43] (Graph Neural Network).

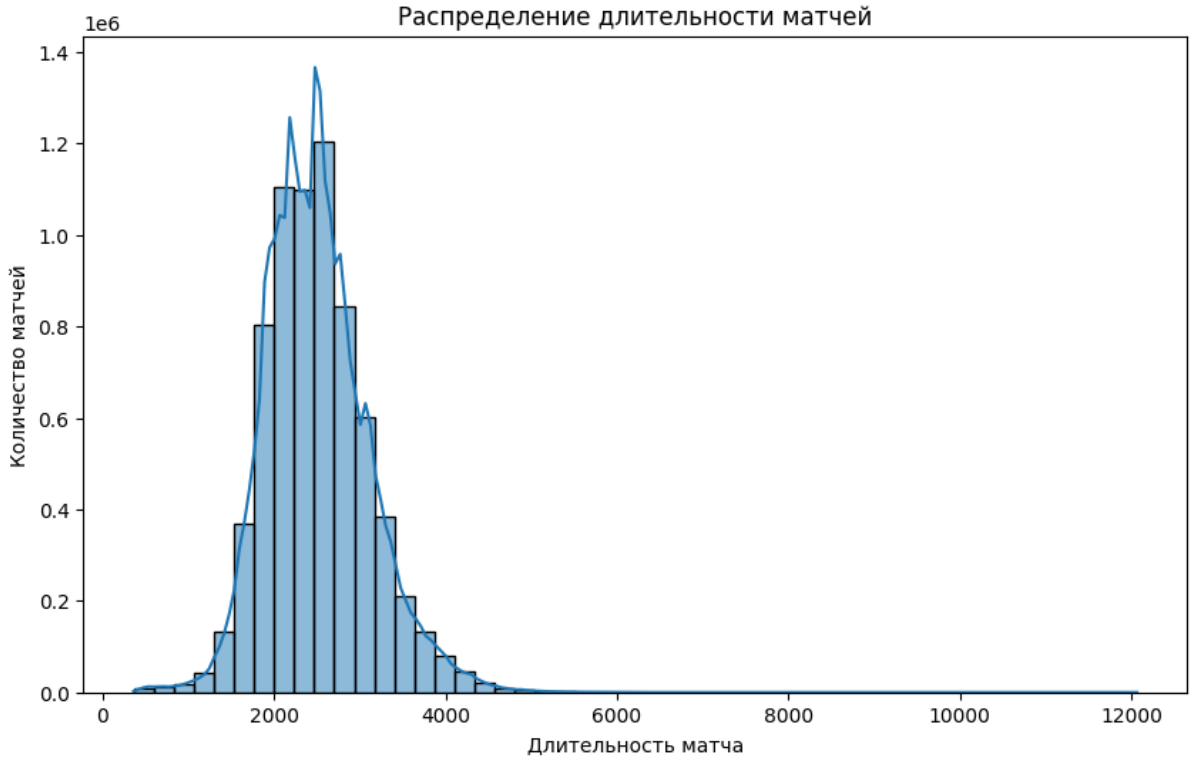

Рис. 7: Гистограмма длительности матчей.

На тренировочном наборе данных модель достигла значения функции потерь (loss) 0.6034 и точности (accuracy) 0.6723 в последней эпохе обучения. На валидационном наборе данных модель показала значение функции потерь 0.7166 и точность 0.5923, результаты представлены в таблице 1.

Модель	Test_Acc	Test_loss	Val Acc	Val_loss
Graph Neural Network	0,6723	0,6034	0,5923	0,7166

Таблица 1: Результат работы модели.

5. Апробация

В данной главе представлено описание опроса фокус-группы, проведенного с целью изучения эффективности и удобства использования платформы игроками Dota 2. Также приведен пример использования платформы для предсказания исхода матчей киберспортивных турниров.

5.1. Опрос фокус-группы

Для проведения апробации разработанной платформы была сформирована фокус-группа, состоящая из 40 опытных игроков в "Dota 2" из четырех стран: России, Марокко, Аргентины, Франции. Участники фокус-группы были подобраны из разных регионов и стран, чтобы покрыть широкий спектр игровых ситуаций и тактик. Кроме того, они имеют разный уровень игрового опыта и умения, что позволяет оценить эффективность платформы в различных условиях.

Процесс апробации заключался в следующем: игроки дожидались, пока в их команде будет выбрано 4 героя, после чего вводили список уже выбранных и заблокированных героев в платформу и получали рекомендацию лучшего выбора. Участники пользовались платформой в течение недели.

Пользователям были заданы следующие вопросы.

1. Насколько полезной для Вас показалась платформа?
2. Насколько удобен был для Вас пользовательский интерфейс?
3. Продолжили бы Вы пользоваться данным сервисом?
4. Как изменился ваш процент побед при использовании платформы?
5. Сколько игр Вы сыграли, используя платформу?

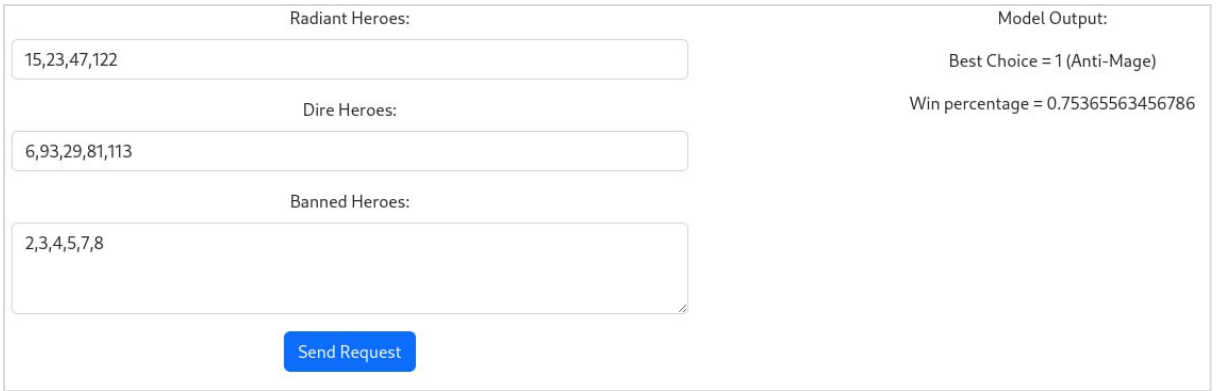

Рис. 8: Пользовательский интерфейс рекомендации лучшего выбора (числа в полях для ввода – идентификационные номера героев).

По окончании тестирования участники отвечали на заранее составленные вопросы. Большое число пользователей отметили значительное повышение процента побед во время пользования платформой, что показано на рисунке 9.

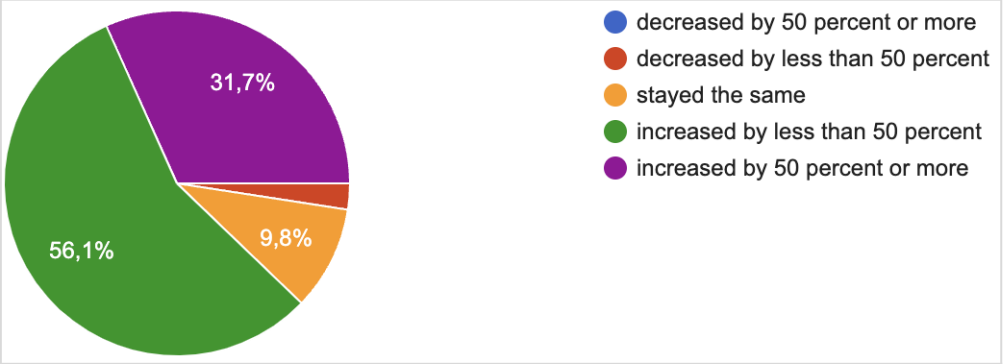

Рис. 9: Диаграмма распределения ответов пользователей на вопрос 4.

Подавляющее большинство пользователей отметили, что готовы были бы дальше использовать платформу в своих играх. Это подтверждается на рисунке 10, где видно, что более 85% участников фокус-группы высоко оценили работу платформы.

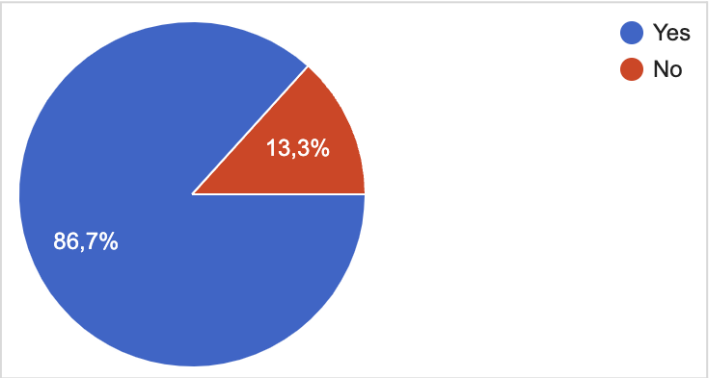

Рис. 10: Диаграмма распределения ответов пользователей на вопрос 3.

Результаты апробации показали, что разработанная платформа является эффективным инструментом для выбора героев в игре "Dota 2".

5.2. Эксплуатация развернутой платформы во время турниров

Платформа была развернута и запущена для проведения тестирования предиктивных моделей, а именно моделей на основе GNN, Logistic Regression [44] и Random Forest [45] на турнирах Lima Major 2023 [46] (22 февраля - 5 марта) и ESL One Berlin Major 2023 [47] (26 апреля - 7 мая).

Перед каждым матчем обоих турниров была собрана предматчевая статистика, которая была преобразована в формат входных данных для предиктивных моделей (подробнее в разделе 3.4.3.). С помощью платформы данные были загружены в модели, и были получены предсказания, которые в дальнейшем были сопоставлены с реальными исходами матчей.

	Lima Major 2023	ESL One Berlin Major
Групповая стадия турнира	52%	47%
Плей-офф турнира	69%	59%
Итого	62%	54%

Таблица 2: Проценты правильных прогнозов платформы в киберспортивных турнирах с использованием модели на основе GNN.

Каждый турнир состоял из групповой и плей-офф стадий. Модели были протестированы на обеих стадиях турниров. На групповой стадии Lima Major 2023 было проведено 149 матча, на плей-офф - 44 матчей. Практически аналогичное соотношение матчей было на ESL One Berlin Major 2023: 147 матчей на групповой стадии и 44 матчей на плей-офф стадии. Результаты тестирования на обоих турнирах различались. Во время анализа матчей групповой стадии процент правильных прогнозов моделей был ниже, чем в матчах плей-офф, что подтверждается в таблице 2 и на рисунке 11.

Предполагается, что понижение процента правильных прогнозов моделей в предсказании матчей ESL One Berlin Major может быть связано с глобальным обновлением 7.33 [48], которое произошло перед турниром и полностью изменило баланс игры. В связи с этим, на момент проведения

турнира не было достаточного количества матчей для обучения, что могло привести к понижению точности предсказаний.

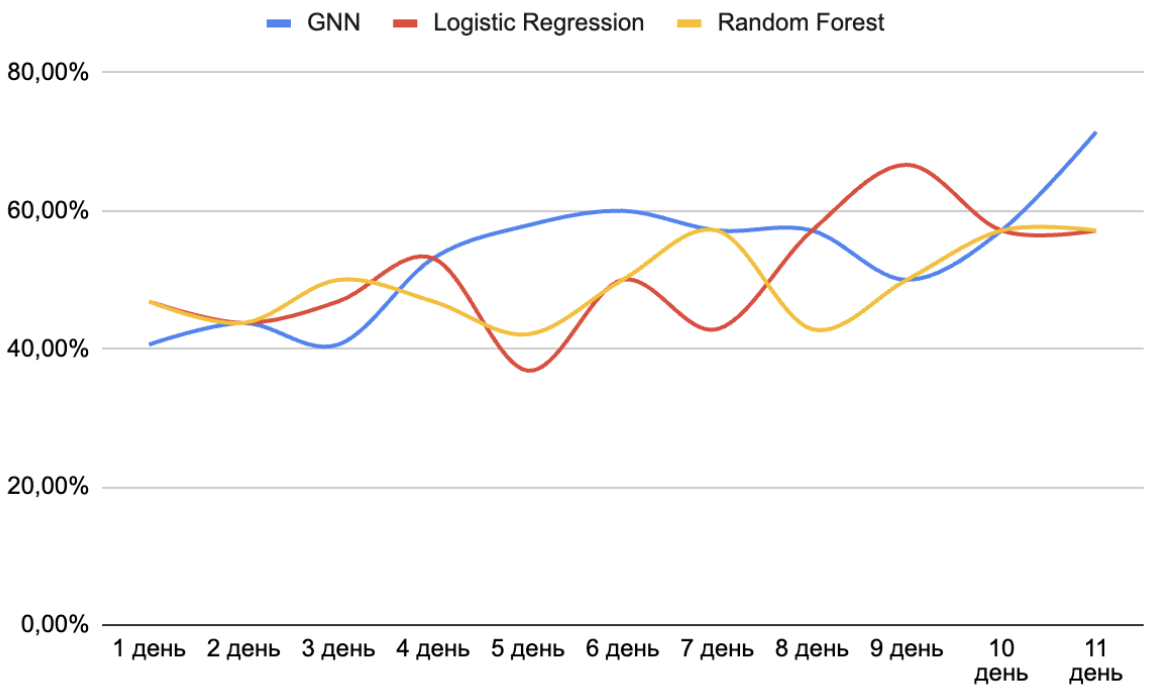

Рис. 11: Диаграмма процентов правильных прогнозов платформы в турнире ESL One Berlin Major 2023.

На рисунках 11 и 12 представлены диаграммы, отображающие распределения процентов правильных предсказаний моделей по ходу действий турниров. Заметно, что с каждым последующим днем турнира модели становятся более точными в предсказании результатов. Это можно объяснить тем, что платформа ежедневно проводит дообучение моделей на новых данных, что улучшает их предиктивные свойства.

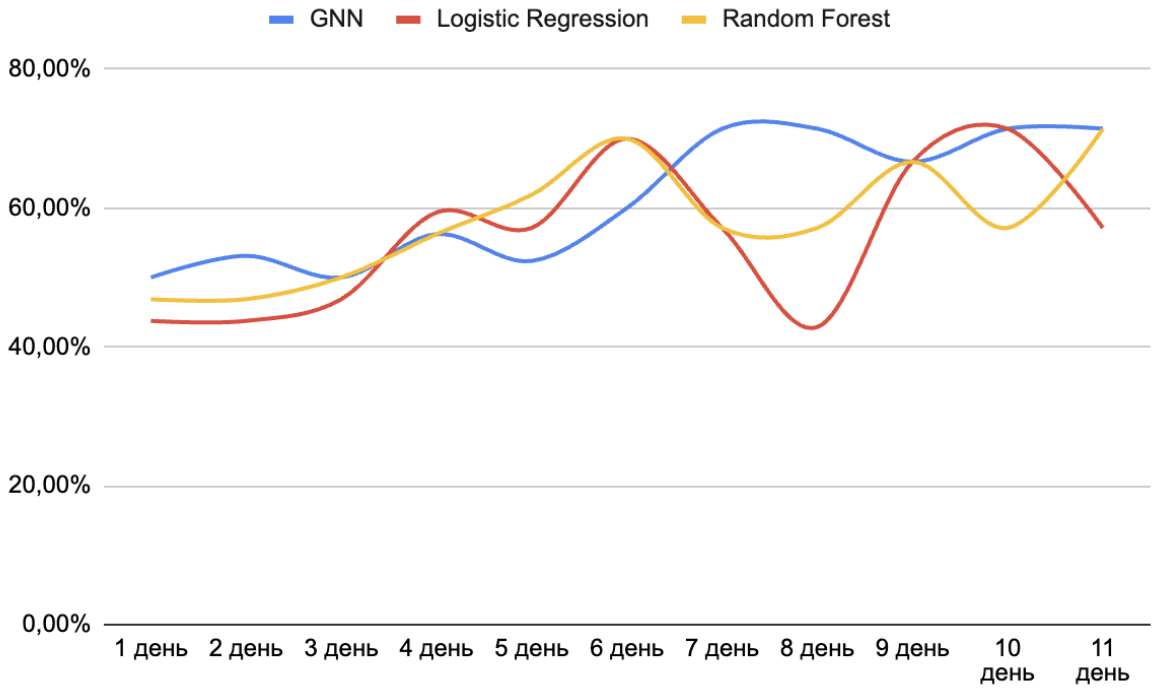

Рис. 12: Диаграмма процентов правильных прогнозов платформы в турнире Lima Major 2023.

6. Заключение

В результате работы были получены следующие результаты.

1. Были проанализированы 11 научных статей по анализу футбольных матчей и применению различных математических моделей, построенных на внутриигровых данных, а также 5 статей по игре "Dota 2" на применение различных моделей машинного обучения к предсказанию матчей. Результаты исследования показали, что использование моделей машинного обучения может дать положительные результаты в предсказании победы одной из команд.

2. Производен сравнительный анализ программных интерфейсов OpenDota API и Steam Web API, который показал необходимость использования OpenDota API из-за детальности получаемых данных о послематчевой статистики.

3. Выполнена реализация платформы предиктивной аналитики для Dota 2 на базе микросервисов:
    - сервис для автоматизированного сбора данных сыгранных матчей, реализованный на языке C# с использованием ASP.NET;
    - сервис для дообучения модели на новых результатах матчей, реализованный на языке Python и использующий pickle как библиотеку сериализации;
    - сервис для предсказания победы по предматчевым данным и для генерации предложений при выборе героев с возможностью дообучения при поступлении новых данных в ежедневном формате, реализованный на языке Python с использованием Fast API фреймворка, а также pickle как библиотеки десериализации;
    - клиентское веб-приложение, получающее от пользователей предматчевые данные и демонстрирующее ответ предиктивной модели в понятном человеку формате, реализованное на языке JavaScript с использованием библиотеки React.

4. Разработана архитектура и спецификация для внешних предиктивных моделей машинного обучения по подключению в

платформу.

5. Разработанная платформа была опубликована в облачной инфраструктуре, а также было настроено непрерывное развертывание с помощью сервиса Github Actions на Linux сервер.

6. Апробация платформы была проведена на фокус-группе, состоящей из 40 человек из четырех стран: России, Марокко, Аргентины, Франции. Результаты обратной связи от участников фокус-группы подтвердили высокую эффективность платформы.

7. Платформа была применена на ежедневной основе для предсказания побед по предматчевой статистике на турнирах Lima Major 2023 и ESL One Berlin Major 2023. Платформа ежедневно проводит дообучение моделей на новых данных, что улучшает их предиктивные свойства.

Список литературы

[1] DFC Intelligence. Global Video Game Consumer Population. URL: https://www.dfcint.com/dossier/global-video-game-consumer-population/

[2] Fariñas, R. A. (2021). Our DotA 2 Olympians. The Reflective Practitioner, 6.

[3] Liquipedia. The International. URL: https://liquipedia.net/dota2/The_International

[4] Dota2.com. URL: https://www.dota2.com/

[5] Dota2.com/esports. URL: https://www.dota2.com/esports

[6] Bányai, Fanni, et al. "The psychology of esports: A systematic literature review." Journal of gambling studies 35 (2019): 351-365.

[7] Boguslavskaya, Vera, et al. "Cybersport community: social structures transformation as a basis for intercultural dialogue." Internet Science: 5th International Conference, INSCI 2018, St. Petersburg, Russia, October 24–26, 2018, Proceedings 5. Springer International Publishing, 2018.

[8] Kashcha, Mariia, Valerii Yatsenko, and Tamás Gyömörei. "Country performance in e-sport: Social and economic development determinants." Journal of International Studies 15.4 (2022).

[9] Pankina, V. V., and R. T. Hadieva. "Cybersport as a phenomenon of the XXI century." Fizicheskaya kul'tura. Sport. Turizm. Dvigatel'naya rekreaciya 1.3 (2016): 34-38.

[10] Palacios-Huerta I. (2004) Structural changes during a century of the world's most popular sport.

[11] Sarmento, Hugo, et al. "Match analysis in football: a systematic review." Journal of sports sciences 32.20 (2014): 1831-1843.

[12] Kahn, Joshua. "Neural network prediction of NFL football games." World Wide Web Electronic Publication 9 (2003): 15.

[13] Ric, Angel, et al. "Dynamics of tactical behaviour in association football when manipulating players' space of interaction." PloS one 12.7 (2017): e0180773.

[14] G. R. Riccardo Ievoli, Lucio Palazzo, “On the use of passing network indicators to predict football outcomes.”

[15] J. B. J. M. Buldu, “Using network science to analyze guardiola’s f.c. barcelona.”

[16] J. B. J. M. Buldu, “Using network science to analyse football passing networks: Dynamics, space, time, and the multilayer nature of the game.”

[17] J. L. P. Tarak Kharrat, Ian G.McHale, “Plus–minus player ratings for soccer.”

[18] S. D. Ian G.McHale, “Identifying key players in soccer teams using network analysis and pass difficulty.”

[19] D. J. Andrew Fast, “The nfl coaching network: Analysis of the social network among professional football coaches.”

[20] B. Gonçalves and D. Coutinho, “Exploring team passing networks and player movement dynamics in youth association football.”

[21] K. Conley and D. Perry, “How does he saw me? a recommendation engine for picking heroes in dota 2.”

[22] A. Agarwala and M. Pearce, “Learning dota 2 team compositions.”

[23] P. S. A. D. R. Gede Adi Aryanata and Y. S. Sudarmojo, “Prediction of dota 2 match result by using analytical hierarchy process method,” 2017.

[24] N. Kinkade and K. yul Kevin Lim, “Dota 2 win prediction,”

[25] K. Akhmedov and A. H. Phan, “Machine learning models for dota 2 outcomes prediction,” 2021.

[26] OpenDota API. URL: https://docs.opendota.com/

[27] Steam Web API. URL: https://steamcommunity.com/dev?l=russian

[28] ASP.NET. URL: https://dotnet.microsoft.com/en-us/apps/aspnet

[29] Entity Framework. URL: https://learn.microsoft.com/en-us/ef/core/

[30] PostgreSQL. URL: https://www.postgresql.org/

[31] Coravel. URL: https://docs.coravel.net/

[32] Quartz.NET. URL: https://www.quartz-scheduler.net/

[33] Fast API. URL: https://fastapi.tiangolo.com/

[34] Pickle python. URL: https://docs.python.org/3/library/pickle.html

[35] React-JS. URL: https://react.dev/

[36] React-JS DOM. URL: https://legacy.reactjs.org/docs/faq-internals.html

[37] Fetch. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

[38] GitHub diploma_spbu. URL: https://github.com/dszharikov/diploma_spbu

[39] Docker. URL: https://docs.docker.com/

[40] Docker-Compose. URL: https://docs.docker.com/compose/

[41] Debian. URL: https://www.debian.org/

[42] GitHub Actions. URL: https://docs.github.com/en/actions

[43] Zhou, Jie, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. "Graph neural networks: A review of methods and applications." AI open 1 (2020): 57-81.

[44] Logistic Regression. URL: https://www.ibm.com/topics/logistic-regression

[45] Random Forest. URL: https://www.ibm.com/topics/random-forest

[46] Lima Major 2023. URL: https://liquipedia.net/dota2/Lima_Major/2023

[47] ESL One Berlin Major 2023. URL: https://www.esl-one.com/berlin/

[48] Dota 2 Gameplay Update 7.33. URL: https://www.dota2.com/patches/7.33