Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем
Направление подготовки /специальность 09.04.04 – Программная инженерия
Направленность (профиль) образовательной программы Управление разработкой программного обеспечения

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____А.В. Бушманов
«_____»_____2022 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: Разработка информационной системы «Прогнозирования рейтинга команд в киберспорте, используя технологию «Большие данные»»

Исполнитель
студент группы 057-ом _____ С. Н. Свечников
(подпись, дата)

Руководитель
доцент, канд. физ.-мат. наук _____ В. В. Еремина
(подпись, дата)

Руководитель научного
содержания программы
магистратуры
профессор, доктор техн. наук _____И. Е. Еремин
(подпись, дата)

Нормоконтроль
Инженер кафедры _____В. Н. Адаменко
(подпись, дата)

Рецензент
Генеральный директор
ООО «Тематика» _____Р. А. Кузьменко
(подпись, дата)

Благовещенск 2022

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем

Утверждаю
Зав. кафедрой
_____А.В. Бушманов
«_____»_____2022 г.

ЗАДАНИЕ

К выпускной квалификационной работе студента Свечникова Степана Николаевича

1. Тема выпускной квалификационной работы: Разработка информационной системы «Прогнозирования рейтинга команд в киберспорте, используя технологию «Большие данные»»
(Приказ №228-уч от 07.02.2022)

2. Срок сдачи студентом законченной работы (проекта) 23.06.2022 г.

3. Исходные данные к выпускной квалификационной работе: предметная область, отчеты по практической подготовке, результаты выступления на научной конференции

4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов): анализ предметной области проводимого исследования, разработка и реализация информационной системы по предсказанию рейтинга команд на основе больших данных

5. Дата выдачи задания: 31.01.2022 г.

6. Руководитель выпускной квалификационной работы: Еремина В.В., доцент, канд. физ.-мат. наук
(фамилия, имя, отчество, должность, уч. степень, уч. звание)

Задание принял к исполнению (31.01.2022)_____

РЕФЕРАТ

Магистерская диссертация содержит 74 с., 35 рисунков, 7 таблиц, 1 приложение, 52 источника.

БОЛЬШИЕ ДАННЫЕ, КИБЕРСПОРТ, ПАРСИНГ, СИНТАКСИЧЕСКИЙ АНАЛИЗ, ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС, МАШИННОЕ ОБУЧЕНИЕ, АЛГОРИТМ, СКРАППИНГ, НЕРЕЛЛЯЦИОННАЯ БАЗА ДАННЫХ.

Целью работы является создание информационной системы с помощью которого можно предсказывать возможного победителя в киберспортивном матче основываясь на технологии Больших данных. Разработанное решение представлено в виде веб-сайта, который может быть доступен на любых устройствах по средствам веб-браузеров.

В данной работе описывается процесс разработки информационной системы, предсказывающей рейтинг команд в киберспорте с помощью технологии Большие данные. Подробно описаны алгоритмы по сбору и обработке полученной информации, а также процесс подготовки машинного обучения. Кроме того, приведен алгоритм взаимодействия серверной и клиентской частей приложения. Взаимосвязь между частями осуществляется с помощью HTTP – протокола.

Задачи выпускной работы:

- анализ предметной области;
- исследование моделей машинного обучения;
- разработка базы данных;
- разработка модели машинного обучения;
- разработка пользовательского интерфейса.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 6

1 ОБЩАЯ ХАРАКТЕРИСТИКА БОЛЬШИХ ДАННЫХ В КИБЕРСПОРТЕ .. 7

1.1 Предметная область 7

1.1.1 Большие данные 7

1.1.2 Киберспорт 11

1.2 Объект исследования 13

1.3 Обзор существующих методов решений исследуемой задачи 14

2 АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ 16

2.1 Алгоритм сбора данных 16

2.1.1 Автоматизированный алгоритм для сбора данных 16

2.1.2 Автоматический алгоритм для сбора данных 21

2.2 Алгоритм подготовки данных для обучения 24

2.2.1 Модель, обученная на прошедших матчах 24

2.2.2 Модель, обученная на расширенном датасете 25

2.3 Обзор возможностей профильного программного обеспечения ... 26

2.4 Обоснование выбора программно-технического обеспечения 29

2.4.1 Выбор языка программирования 29

2.4.2 Выбор базы данных 32

3 ОСНОВНЫЕ ЭТАПЫ ПРАКТИЧЕСКОЙ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА 35

3.1Проектирование базы данных 35

3.2 Разработка алгоритмов для сбора данных 41

3.3 Разработка алгоритмов для обработки данных 47

3.4 Разработка алгоритмов для создания модели машинного обучения 48

3.3.1 Классическое обучения 50

3.3.2 Обучение с подкреплением 50

3.3.3 Ансамблевые методы 51

3.3.4 Нейросети и глубокое обучение 51

3.3.5 Разработка решения поставленной задачи 52

3.5 Разработка серверной части 56

3.6 Разработка клиентской части 58

3.7 Результаты фактического тестирования программного продукта 63

ЗАКЛЮЧЕНИЕ 66

БИБЛИОГРАФИЧЕСКИЕ ССЫЛКИ 68

БИБЛИОГРАФИЧЕСКИЙ СПИСОК 69

ПРИЛОЖЕНИЕ А 74

ВВЕДЕНИЕ

В настоящее время все сферы человеческой жизни производят большие объемы многообразных данных. Все эти данные могут быть использованы для анализа и нахождения определенных зависимостей в них.

В последние годы киберспорт все более тесно входит в нашу жизнь. В мире ежегодно проводятся сотни соревнований различной степени мастерства. Все эти мероприятия создают огромные массивы данных, проанализировав которые можно выдавать командам полезные советы по улучшению игровой тактики, получения возможного победителя встречи и другую игровую статистику. Именно в качестве инструмента для анализа, поиска взаимосвязей и дальнейшего получения результатов выступает технология Больших данных.

Объектом исследования является информационная система предсказания рейтинга команд в киберспорте с помощью технологии Большие данные.

Предметом исследования является методика разработки алгоритмов по сбору и обработке данных, обучения моделей машинного обучения и взаимодействия пользователя через интерфейс с обученной моделью машинного обучения.

Целью работы является создание информационной системы, содержащей автономно работающие скрипты по сбору и обработке информации и обучению моделей машинного обучения и разработка интерфейса взаимодействия пользователя с моделью машинного обучения с помощью веб-сайта.

Для выполнения поставленных целей необходимо решить следующие задачи:

- анализ предметной области;
- исследование моделей машинного обучения;
- разработка базы данных;
- разработка модели машинного обучения;
- разработка пользовательского интерфейса.

1 ОБЩАЯ ХАРАКТЕРИСТИКА БОЛЬШИХ ДАННЫХ В КИБЕРСПОРТЕ

1.1 Предметная область

1.1.1 Большие данные

В 2001 году компания Gartner предложила следующее определение (которое до сих пор применяется): Большие данные - это разнообразные данные, которые поступают с постоянно растущей скоростью и объем которых постоянно растет. Таким образом, три основных свойства больших данных - разнообразие, высокая скорость поступления и большой объем (рис. 1).

Рисунок 1 – Свойства больших данных

Если говорить простыми словами, большие данные – это необычайно большие и сложные наборы данных, обычно из нестандартных источников. Размер этих наборов данных настолько велик, что традиционные программы для обработки не могут с ними справиться. Зато большие данные можно использовать для решения бизнес-задач, которые раньше казались слишком

сложными.

Основные свойства больших данных:

– Объем

Количество данных – важный фактор. Располагая ими в больших количествах, Вам потребуется обрабатывать большие объемы неструктурированных данных низкой плотности. Ценность таких данных не всегда известна. Это могут быть данные каналов Twitter, данные посещаемости веб-страниц, а также данные мобильных приложений, сетевой трафик, данные датчиков. В некоторые организации могут поступать десятки терабайт данных, в другие - сотни петабайт.

– Скорость

Скорость в данном контексте - это скорость приема данных и, возможно, действий на их основе. Обычно высокоскоростные потоки данных поступают прямо в оперативную память, а не записываются на диск. Некоторые "умные" продукты, функционирующие на основе Интернета, работают в режиме реального или практически реального времени. Соответственно, такие данные требуют оценки и действий в реальном времени.

– Разнообразие

Разнообразие означает, что доступные данные принадлежат к разным типам. Традиционные типы данных структурированы и могут быть сразу сохранены в реляционной базе данных. С появлением BigData данные стали поступать в неструктурированном виде. Такие неструктурированные и полуструктурированные типы данных как текст, аудио и видео, требуют дополнительной обработки для определения их значения и поддержки метаданных.

Рисунок 2 – Разнообразие больших данных

За прошедшие годы у больших данных появились две дополнительных характеристики: ценность и достоверность. Данные имеют внутренне присущую им ценность. Однако, чтобы они приносили пользу, эту ценность необходимо раскрыть.

Рисунок 3 – Характеристики больших данных

Новейшие достижения в сфере технологий позволили значительно снизить стоимость хранилищ и вычислений, что дает возможность хранить и обрабатывать постоянно растущие объемы данных. Современные технологии позволяют хранить и обрабатывать больше данных за меньшую стоимость.

Преимущества больших данных:

– Большие данные дают возможность получать более полные ответы, поэтому они предоставляют больше информации.

– Более подробные ответы означают, что Вы можете быть уверены в достоверности данных, - что обеспечивает абсолютно новый подход к решению задач [1].

Для анализа больших объемов структурированной и неструктурированной информации необходимо выполнять в несколько этапов:

– чистка данных (data cleaning) – поиск и исправление ошибок в первичном наборе информации, например, ошибки ручного ввода (опечатки), некорректные значения с измерительных приборов из-за кратковременных сбоев и т.д.;

– генерация предикторов (feature engineering) – переменных для построения аналитических моделей, например, образование, стаж работы, пол и возраст потенциального заемщика;

– построение и обучение аналитической модели (model selection) для предсказания целевой (таргетной) переменной. Так проверяются гипотезы о зависимости таргетной переменной от предикторов. Например, сколько дней составляет просрочка по кредиту для заемщика со средним образованием и стажем работы менее 3-х месяцев.

В современной практике существует большое количество различных методов для сбора и анализа. Но к основным методам сбора и анализа больших данных относят следующие:

– Data Mining - обучение ассоциативным правилам, классификация, кластерный и регрессионный анализ;

– краудсорсинг - категоризация и обогащение данных народными силами, т.е. с добровольной помощью сторонних лиц;

– смешение и интеграция разнородных данных, таких как, цифровая обработка сигналов и обработка естественного языка;

– машинное обучение (Machine Learning), включая искусственные нейронные сети, сетевой анализ, методы оптимизации и генетические алгоритмы;

– распознавание образов;

– прогнозная аналитика;

– имитационное моделирование;

– пространственный и статистический анализ;

– визуализация аналитических данных — рисунки, графики, диаграммы, таблицы.

Программно-аппаратные средства работы с Big Data предусматривают масштабируемость, параллельные вычисления и распределенность, т.к. непрерывное увеличение объема – это одна из главных характеристик больших данных. К основным технологиям относят нереляционные базы данных (NoSQL), модель обработки информации MapReduce, компоненты кластерной экосистемы Hadoop, языки программирования R и Python, а также специализированные продукты Apache (Spark, AirFlow, Kafka, HBase и др.).

1.1.2 Киберспорт

Киберспорт – это соревнования по различным компьютерным играм. Матчи могут быть командными или одиночными в зависимости от особенностей дисциплины.

В ключевую аудиторию киберспорта входят люди от 15 до 25 лет, то есть те, кто сам может стать киберспортсменом, а также является потребителем игровых девайсов и других товаров, которые рекламируются крупными компаниями во время проведения чемпионатов (рис. 4).

История киберспорта начинается практически сразу после появления компьютерных игр.

Одно из первых довольно крупных соревнований было проведено в 1980 году, когда Atari решила провести чемпионат по Space Invaders. Мероприятие было довольно масштабным, ведь в нем участвовало 10 000 человек.

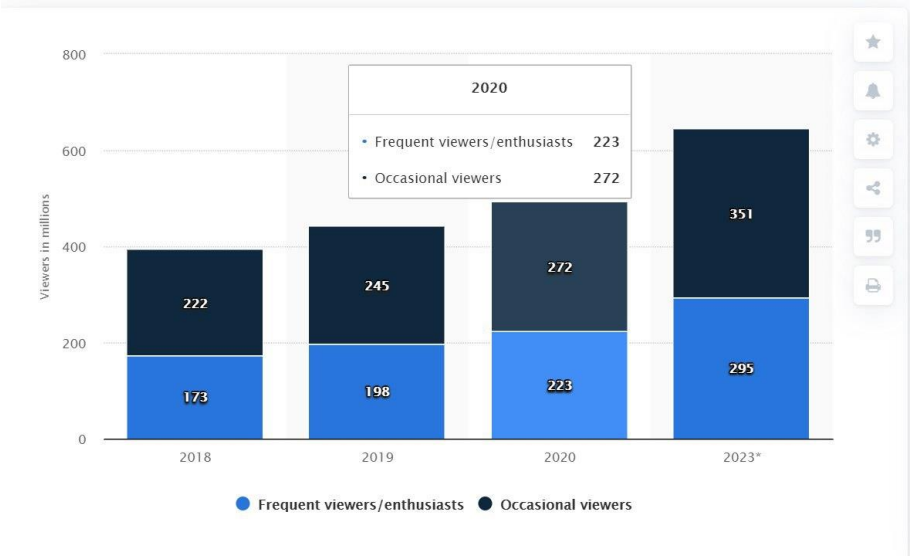

Рисунок 4 – График роста аудитории

Но именно о развитии киберспорта можно было говорить после того, как начались турниры по Quake, ведь игра выглядела действительно соревновательно, а разработчик предлагал победителю подарить собственное Феррари, что в итоге и сделал.

На данный момент киберспорт признан не во всех странах, но при этом большинство, как минимум, участвует в киберспортивной жизни, принимая у себя различные чемпионаты.

Призовые турниров ежегодно увеличиваются в большинстве игр, а сам киберспорт постепенно признается официальным видом спорта во все большем количестве стран.

Помимо этого, во многих странах введен отдельный налог на выигрыш, полученный с киберспортивных состязаний [2].

Киберспортивные правила являются отдельными для каждой игры. Регламент проведения чемпионатов устанавливается только самими организаторами, а также может устанавливаться разработчиком игры.

В некоторых странах устанавливаются определенные условия, которым должны соответствовать турниры.

Без сомнений киберспорт для детей и взрослых будет развиваться семимильными шагами и дальше.

Несмотря на то, что в последнее время рынок мобильных игр забирает большую часть аудитории, призовые в киберспортивных соревнованиях по-прежнему увеличиваются, и неизвестно, когда этот рост прекратится.

Пока все больше людей приобщается к профессиональному гейминг, интересуется матчами и различными игровыми девайсами, поэтому многие компании по-прежнему заинтересованы в организации чемпионатов.

1.2 Объект исследования

Объектом проводимого исследования является проектирование и разработка информационной системы с применением технологии больших данных в киберспорте.

Проблема исследования – теоретически проанализирована научная и методическая литература по проблеме разработки системы по прогнозированию.

Предмет исследования – проектирование и разработка информационной системы для предсказания рейтинга команд в киберспорте.

Цель исследования – изучение и практическое применение методов проектирования и разработки системы для предсказания.

Результатом разработки будет система, которая позволит благодаря собранным данным с сайта, содержащего статистику матчей, предсказывать рейтинг команд в киберспорте. В данной работе описаны основные этапы работы над программным продуктом и методы, которые были выбраны в качестве основных при обучении модели на основе больших массивов информации.

1.3 Обзор существующих методов решений исследуемой задачи

Существует большое количество разнообразных площадок на которых публикуются результаты будущих или текущих матчей по различным кибер-спортивным дисциплинам.

Они строят свои прогнозы основываясь на уже прошедших матчах. Все данные анализируются в «ручном» режиме. То, есть за предсказанный исход того, или иного матча отвечает аналитик или группа аналитиков.

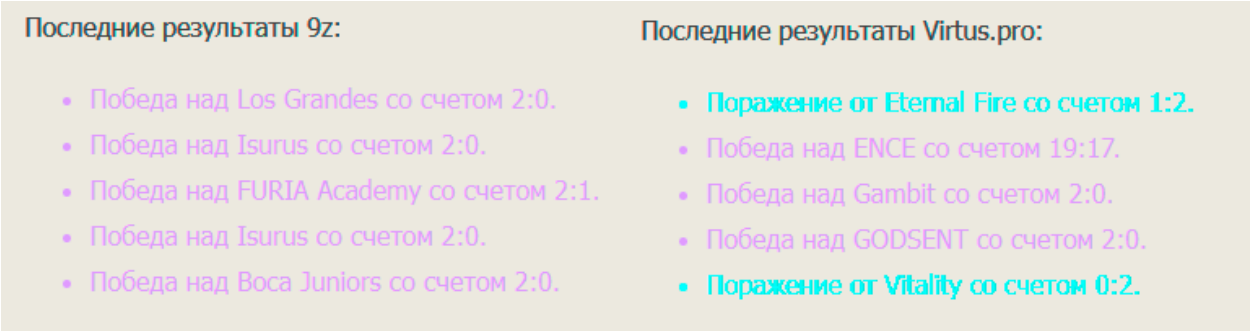

Рисунок 5 – Анализируемые данные

Помимо публикуемого результата, зачастую также имеется и анализ формы команд, статистика встреч двух команд.

Также посетители могут проголосовать за предоставляемый им прогноз.

Рисунок 6 – Краткая форма результатов

Но есть и независимые от площадок аналитики, которые публикуют свои результаты в различных социальных сетях, мессенджерах. Они также собирают и анализируют данные, только как правило у таких пользователей количество предоставляемой информации для общественности гораздо меньше. Это связано с тем, что они сами занимаются всеми шагами для получения результата.

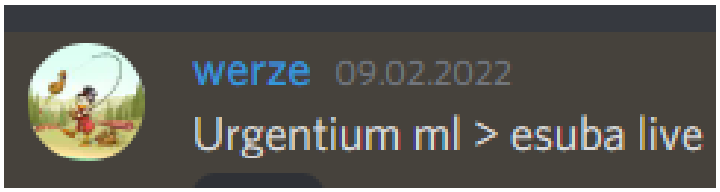

Рисунок 7 – Предполагаемый результат, опубликованный на канале в Discord

Конечно не стоит сбрасывать со счетов, что как правило такие аналитики, имеют техническое образование и имеют минимальные навыки в программирование. С помощью которых, могут облегчить себе жизнь написанием несложных скрапперов данных по определённым сайтам.

И как правило они не обременены рамками площадок, в отличие от первых. То есть, одиночки сами выбирают формат, в котором они будут писать свои результаты для общественности, в то время как аналитики на площадках обязаны предоставлять свои результаты в соответствии с регламентом, того или иного сайта, на котором будут располагаться их предсказания.

2 АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ

Для решения поставленной задачи необходимо наличие заранее обученной модели машинного обучения или набора данных для обучения в текущий момент и интерфейса пользователя для наилучшего пользовательского опыта. Модель машинного обучения - это приложение искусственного интеллекта, которое дает возможность автоматически учиться и совершенствоваться на основе собственного опыта без явного участия человека.

Для сбора данных используются автоматизированные алгоритмы по веб-скрапингу и парсингу данных из сети интернет. Веб-скрапинг - это технология получения веб-данных путем извлечения их со страниц веб-ресурсов. Парсинг – это принятое в информатике определение синтаксического анализа.

Языком программирования выступает Python, который является одним из ведущих в направлении Data Science.

2.1 Алгоритм сбора данных

На данный момент в разработанной информационной системе существует два алгоритма для сбора данных – автоматизированный, для запуска которого необходимо участие человека и автоматический, запускаемый по расписанию.

2.1.1Автоматизированный алгоритм для сбора данных

Данный алгоритм необходим для первичной выборки данных. Его функционал разбит на 5 последовательно запускаемых функции.

Первым является скраппер сайта с данными. Ему на вход передаются: ссылка для сбора данных, файл в который необходимо сохранить собранные данные и необязательный атрибут для записи в файл.

При запуске данной функции первым делом отправляется запрос на страницу по ссылке, указанной при вызове, также для корректного запроса передаются http-заголовки. Заголовки HTTP позволяют клиенту и серверу отправлять дополнительную информацию с HTTP запросом или ответом. Если

запрос не вернет «код ответа 200», то работа будет завершена. Иначе, будет продолжена работа на данной странице.

Будет создан или открыт уже имеющийся выходной файл, после создастся объект для работы с разметкой текущей страницы, из которого уже и будет браться вся информация. На данном этапе нас интересует лишь статистика с данной страницы, которая храниться в теге <tbody></tbody>, его и записываем в файл. После чего проверяем имеется ли на странице переход на следующую страницу, если имеется, то переходим на следующую страницу, тем самым создаем рекурсию данной функции, в противном случае завершаем работу данного модуля (рис. 8).

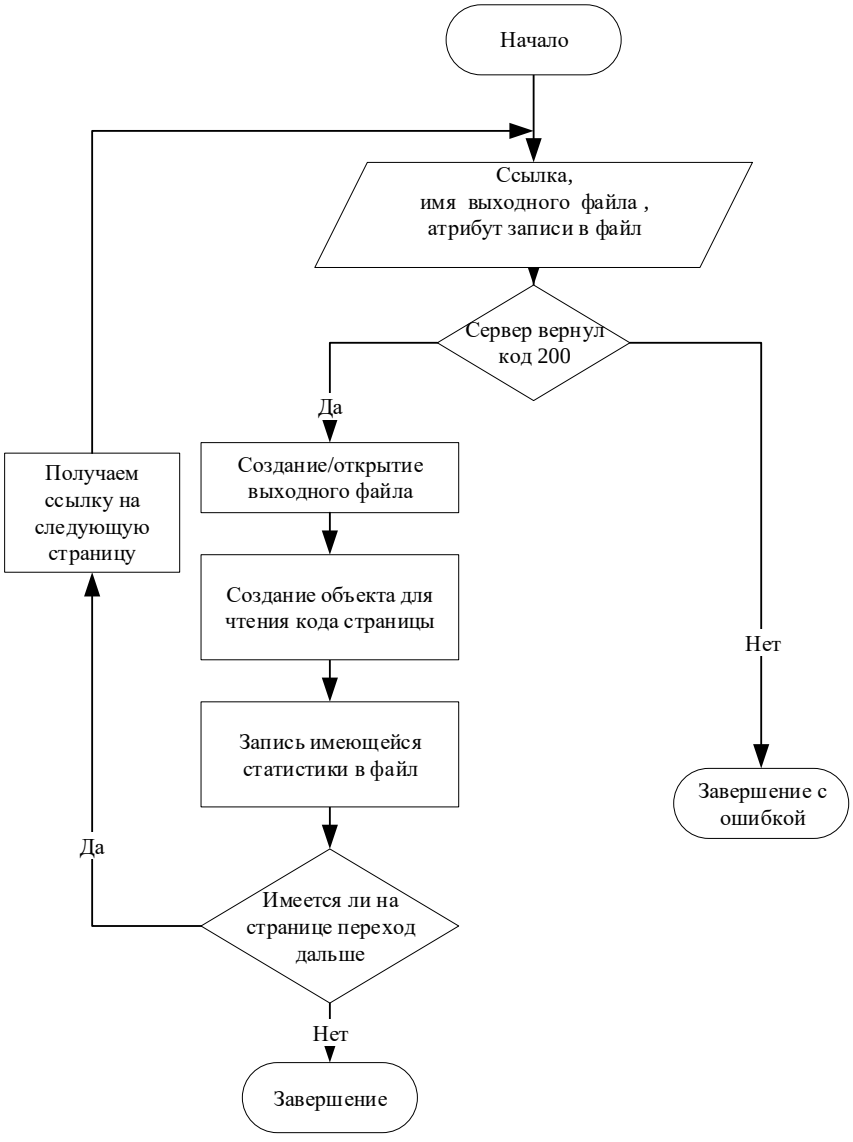

Рисунок 8 – Алгоритм работы скраппера

Следующим в цепочке вызова функций является парсер. Он выполняет обработку собранных скраппером данных и записывает их в указанный файл с расширением .csv.

Функция начинает свою работу со считывания входного файла и создания объекта библиотеки BeautifulSoup, необходимой для синтаксического анализа веб-страниц, передачей в качестве одного из параметров данных файла. После этого следует два вложенных друг в друга цикла, перебирающие элементы.

Первый цикл перебирает собранные раннее теги <tbody>, а внутренний совершается по вложенным в перебираемом элементе тегам <tr>, именно он и является главным в данном цикле. После получаем ссылку на матч и тут же совершаем запрос на этот адрес, в случае если сервер вернул код, отличный от 200, то завершается работа функции. Если был возвращен 200, то создается новый объект по текущей странице, с которой получаем название команды и игроков. Получив данные со страницы, обращаемся к данным из текущего элемент tr, из которого получаем игровую локацию, заносим собранные данные в переменную.

По завершению внешнего цикла происходит запись собранных данных в «выходной файл», имя которого было указана при вызове функции.

Алгоритм работы в виде блок-схемы представлен в приложении А.

После успешного выполнения парсера вызывается процесс получения текущих команд, в процессе которого происходит получение подробных данных о текущих составах текущих игровых команд.

Алгоритм работы представлен на рисунке 9. Поясним шаги работы данной функции.

Функция принимает один параметр – массив файлов с данными. Данные файлы поочередно считываются в переменную, которая дальше используется для создания объекта BeautifulSoup. После этого следует 3 вложенных друг в друга. В третьем цикле первым делом происходит получение ссылки из ранее

созданного объекта. Получив ссылку, происходит запрос по ней и в случае получения 200 кода ответа начинается работа по обработке текущей страницы. На этом этапе программа собирает такие данные, как ссылка на команду, название команды, логотип команды, информация о ростере команды (ссылка на профиль игрока, никнейм, фотография).

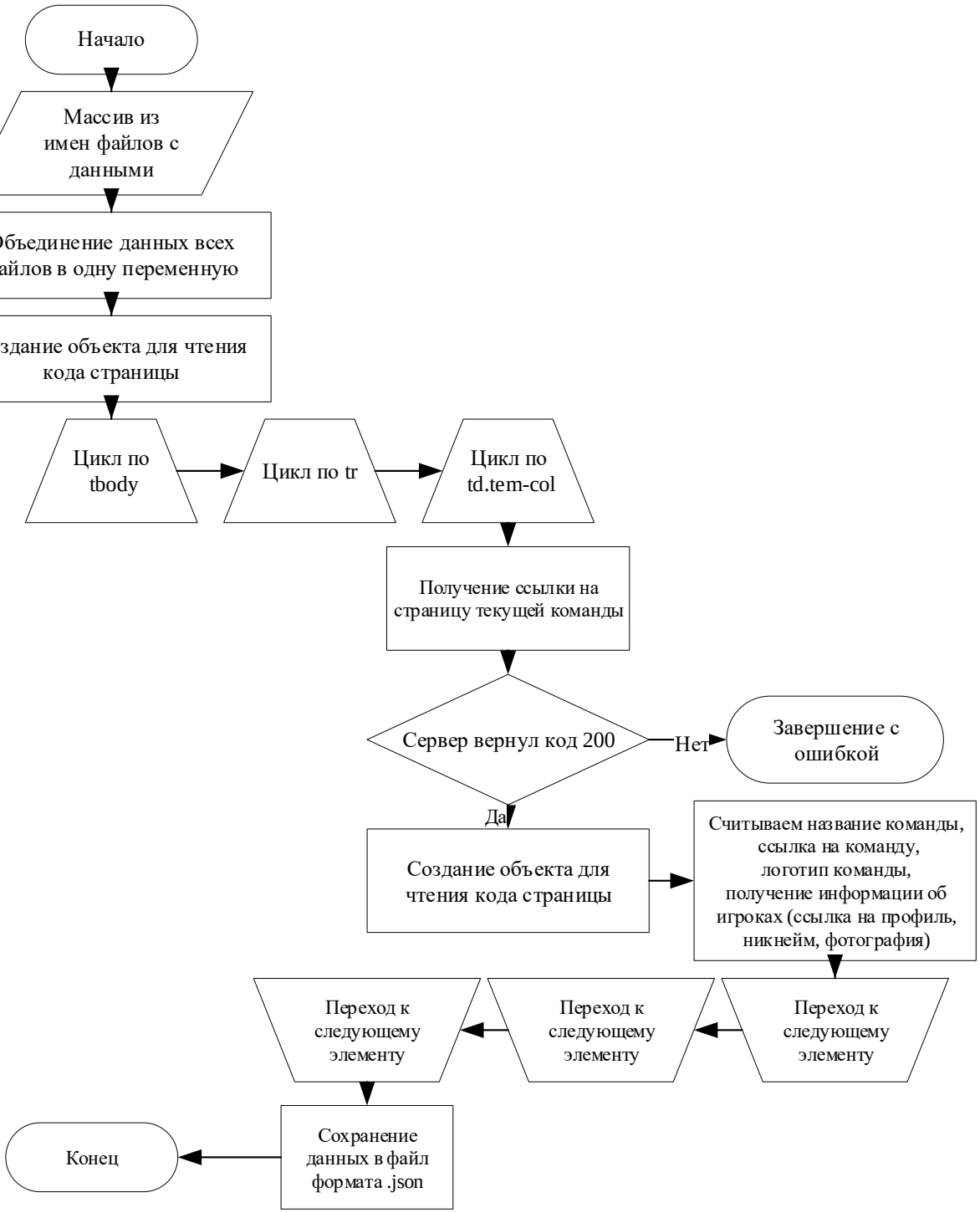

Рисунок 9 – Получение данных о текущих командах

Распарсив собранные данные и собрав данные команд теперь необходимо приступать к очистке собранных данных. Для этой цели имеется отдельная функция. Которая принимает несколько параметров, первый – массив файлов, вышедших после парсинга, второй – файл со списком команд, последний – имя выходного файла (рис 10.).

На этом этапе удаляются не удовлетворяющие нас данные, то есть команд в которых нет на данный момент игровых составов.

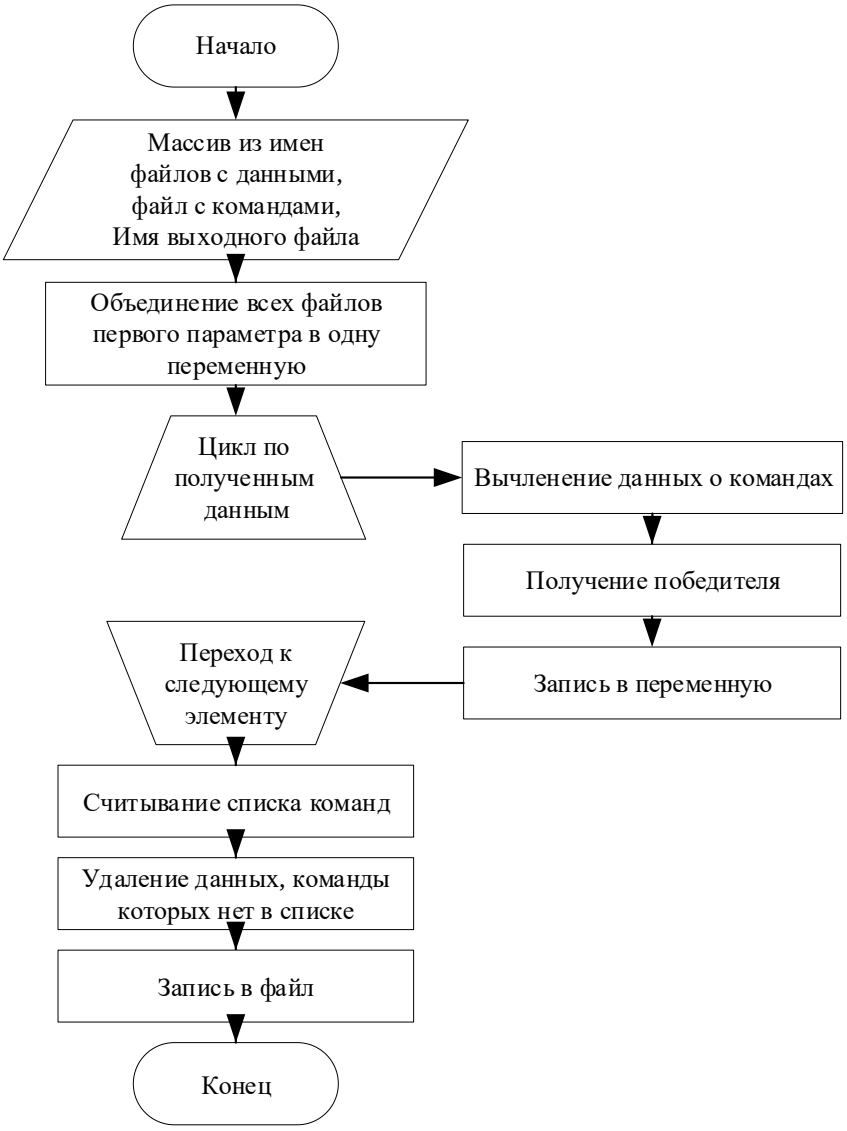

Рисунок 10 – Блок-схема алгоритма рефакторинга

Первым делом данные файлов объединяются в одну переменную, по которой идет цикл. В цикле на каждой итерации из собранных данных получаются данные о командах, выясняется кто победил в том или ином матче, из

всех этих данных формируется объект, записываемый в переменную. По окончанию цикла происходит удаление данных, в которых указаны команды, не присутствующие в файле со списком команд.

После очистки, оставшиеся данные записываются в файл.

Последний этап – внесение в базу данных. Первым делом необходимо создать подключение. Так как, база данных находится на выделенном сервере, то и время подключение, и время отклика зависит от стабильности интернет-соединения как у клиента, так и у сервера базы данных. Подключившись, необходимо указать имя базы данных с которой будет вестись работа.

Дальше последовательно загружаются данные из подготовленных файлов, в соответственные коллекции (рис. 11).

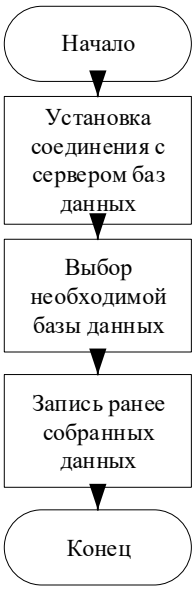

Рисунок 11 – Алгоритм записи в базу данных

2.1.2Автоматический алгоритм для сбора данных

Данный алгоритм является улучшенной версией автоматизированного алгоритма, при этом не требует участия человека в своей работе. Запуск производится по расписанию – каждую субботу. Время работы напрямую зависит от количества новых данных. Данный алгоритм ведет логирование в базе данных о начале и конце работы, об обработанных данных для избегания дублирования информации и конечно же вносит собранную информацию.

Работа алгоритма начинается с установления соединения с сервером баз данных и выбором необходимой базы данных. После подключения, программа обращается к коллекции, в которой хранятся хэши записанных раннее данных и берет последний добавленный, он будет необходим для работы алгоритма в дальнейшем. После того, как была получена последняя запись в коллекции, производится запись о начале парсинга в другую коллекцию, внеся новый документ в коллекцию начинается процесс по сбору и форматированию данных.

Процесс работы данного алгоритма включает в себя части различных этапов автоматизированного алгоритма по сбору данных.

Первым делом необходимо отправить запрос на страницу и в случае, если вернется удовлетворительный код, то продолжаем работу, путем создания объекта BeautifulSoup. После чего происходит проверка на наличие таблицы с данными на странице, если все в порядке, то находим все вхождения тега <tr> в эту таблицу по которым запускаем перебор. Теперь работая со строками таблицы, необходимо получить ссылку на подробную страницу с текущим матчем. Получив ссылку, проверяем, что хэш текущей строки и полученный ранее не равны, и что такого хэша нет в базе данных.

Пройдя данные проверки, делаем запрос на данный адрес, и также, в случае если сервер вернет код не равный 200, то прекращаем работу скрипта, иначе получаем информацию о командах, учувствовавших во встрече (название и состав).

После работы со страницей, скрипт переходит к работе с текущим элементом цикла из которого получает данные о победителе, путем сравнения набранного количества очков и игровой локации, на которой проходила встреча.

Собрав все данные о текущей встрече, происходит их запись в базу данных. После чего записывается хэш ссылки на подробную страницу текущего матча.

Пройдя по всем строкам таблицы со страницы, проверяется имеется ли на странице возможность перейти на следующую, в случае если нет такой

возможности, то скрипт завершает свою работу. Иначе, происходит перенаправление на следующую страницу с задержкой в 3 секунды (чтобы избежать слишком частых запросов на сайт). Таким, образом этот процесс уходит в рекурсию до тех пор, пока не будет выполнено одно из условий, либо не будет возможности перейти на следующую страницу, либо встретится элемент, уже записанный в базе данных.

Завершив работу по сбору данных, в базу данных вносится дата и время завершения скрипта.

Описанный выше алгоритм представлен на рисунке 12.

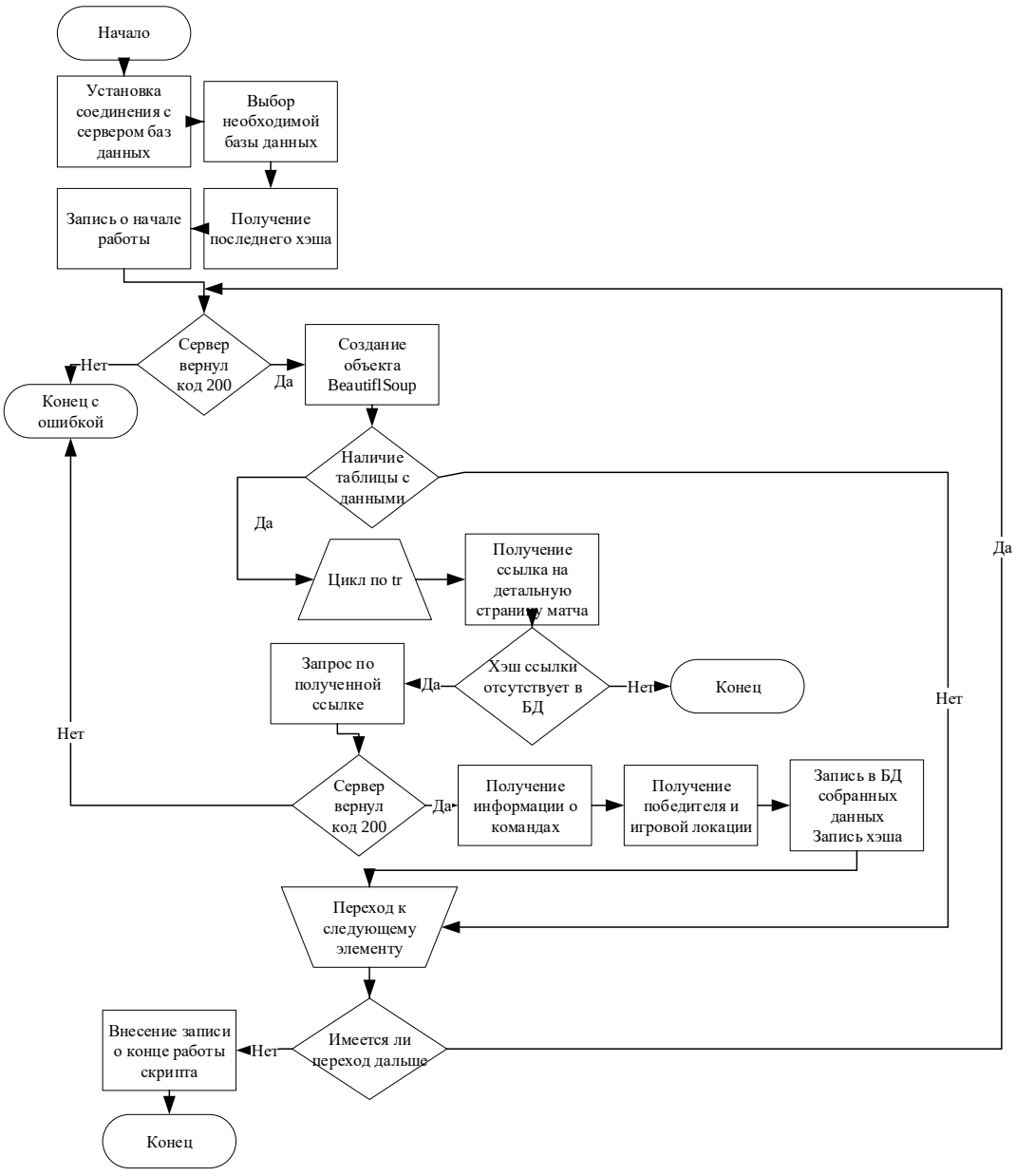

Рисунок 12 – Алгоритм работы автоматического парсера

2.2 Алгоритм подготовки данных для обучения

В реализованном программном продукте существует две модели машинного обучения. Первая обучена на собранных в коллекции данных, в которой хранятся данные о победителе и участниках встречи. Вторая обучается на немного схожих данных, только для их подготовки необходимо провести еще манипуляции по получения данных.

Для наилучшего опыта использования информационной системой, модели после обучения сохраняются, чтобы не время пользователя и не переобучаться каждый раз заново, так как этот процесс занимает длительное время.

2.2.1 Модель, обученная на прошедших матчах

Так как все данные хранятся удаленно, на сервере, то необходимо первым делом получить эти самые данные. Для этого устанавливаем соединение с сервером базы данных и указываем необходимую нам для работы базу данных. После чего, считываем данные с коллекции в переменную.

Считав данные приступаем к разбиению данных на выборки, необходимые для обучения модели и ее последующего тестирования. Выполнив данный этап необходимо приступать к обучению модели. Этот процесс также включает в себя проверку модели на тестовой выборке. Обучив и протестировав теперь ее необходимо сохранить, для дальнейшего использования.

Алгоритм работы представлен на рисунке 13, выполненный в виде блок-схемы.

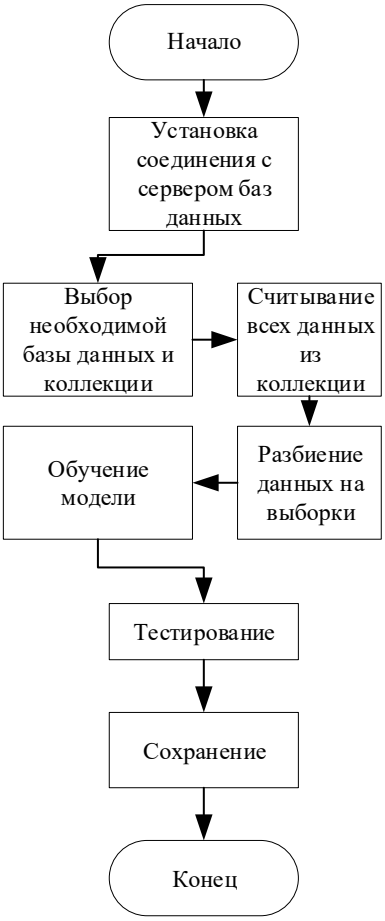

Рисунок 13 - Алгоритм обучения модели на прошедших матчах

2.2.2Модель, обученная на расширенном датасете

В собранной выборке данных присутствуют команды с их составами, где у каждого из игроков указан его никнеймы, фотография и ссылка на профиль с данными. Исходя из этих данных, можно собрать дополнительную статистку по каждому игрок.

Для этого необходимо запустить скрипт, который будет проходить по каждой записи в коллекции с командами и запускать веб-скраппер с парсером на вход которому поступает ссылка на профиль игрока, о котором необходимо получить информацию. Получив информацию, она сразу добавляется в базу данных.

Данный скрипт нет необходимости запускать часто, частота запуска не чаще чем 1 раз в месяц в ручном режиме, данный скрипт также запускается по расписанию.

Получив данные об игроках и их текущей статистике, необходимо приступать к расширению данных и вынесению их в отдельную коллекцию, для того чтобы разные модели использовали разные данные, и при обучении текущей модели каждый раз не тратилось время на получение данных из разных таблиц.

Для этого после завершения предыдущего этапа, в автоматическом режиме запускается скрипт, который позволяет получить данные со средними арифметическими показателями игроков по команде. В итоговом виде таблица будет хранить такие показатели игроков, как процент раундов в которых игрок сделал что либо, среднее значение нанесенного урона, значение их вклада в игру и другие (рис. 14).

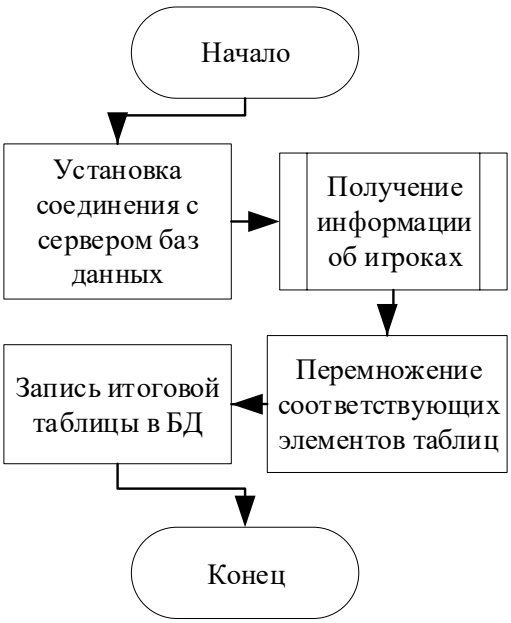

Рисунок 14 – Алгоритм подготовки расширенного набора данных

Далее происходит процесс обучения, который не отличается от описанного в пункте 3.2.1, кроме как используемыми данными.

2.3 Обзор возможностей профильного программного обеспечения

PyCharm – это интеллектуальная Python IDE с полным набором средств для эффективной разработки на языке Python. Выпускается в двух вариантах – бесплатная версия PyCharm Community Edition и поддерживающая больший

набор возможностей PyCharm Professional Edition. PyCharm выполняет инспекцию кода на лету, автодополнение, в том числе основываясь на информации, полученной во время исполнения кода, навигацию по коду, обеспечивает множество рефакторингов.

Таблица 1 – Сравнение версий «PyCharm»

	PyCharm Professional Edition	PyCharm Community Edition
Функциональный редактор Python	+	+
Инструмент запуска тестов и графический отладчик	+	+
Навигация по коду и рефакторинги	+	+
Инспекции кода	+	+
Поддержка систем контроля версий	+	+
Инструменты для научных вычислений	+	
Веб-разработка	+	
Веб-фреймворки Python	+	
Python-профилировщик	+	
Возможности удаленной разработки	+	
Поддержка баз данных и SQL	+	

К ключевым особенностям можно отнести следующие:

- Мощный и функциональный редактор кода с подсветкой синтаксиса, авто-форматированием и авто-отступами для поддерживаемых языков.
- Простая и мощная навигация в коде.
- Помощь при написании кода, включающая в себя автодополнение, авто-импорт, шаблоны кода, проверка на совместимость версии интерпретатора языка, и многое другое.
- Быстрый просмотр документации для любого элемента прямо в окне редактора, просмотр внешеней документации через браузер, поддержка docstring – генерация, подсветка, автодополнение и многое другое.
- Большое количество инспекций кода.
- Мощный рефакторинг кода, который предоставляет широкие возможности по выполнению быстрых глобальных изменений в проекте.
- Полная поддержка свежих версий Django фреймворка.
- Поддержка Google App Engine.
- Поддержка IronPython, Jython, Cython, PyPy, PyQt, и др.
- Редактор Javascript, Coffescript, HTML/CSS, SASS, LESS, HAML.
- Интеграция с системами контроля версий (VCS).
- UML диаграммы классов, диаграммы моделей Django и Google App Engine.
- Интегрированное Unit тестирование.
- Интерактивные консоли для Python, Django, SSH, отладчика и баз данных.
- Полнофункциональный графический отладчик (Debugger).
- Поддержка схем наиболее популярных IDE/редакторов. таких как Netbeans, Eclipse, Emacs, эмуляция VIM редактора.
- PyCharm имеет несколько цветовых схем, а также настраиваемую подсветку синтаксиса кода.
- Интеграция с баг/issue-треккерами, такими как JIRA, Youtrack, Lighthouse, Pivotal Tracker, GitHub, Redmine.
- Огромная, постоянно пополняемая коллекция плагинов.

– Кроссплатформенность (Windows, Mac OS X, Linux).

На рисунке представлено окно по созданию нового проекта. Как можно видеть слева перечислены различные поддерживаемые фреймворки, начиная от консольных приложений на чистом Python, заканчивая настольными приложениями и веб-сервисами в связке с Java Script.

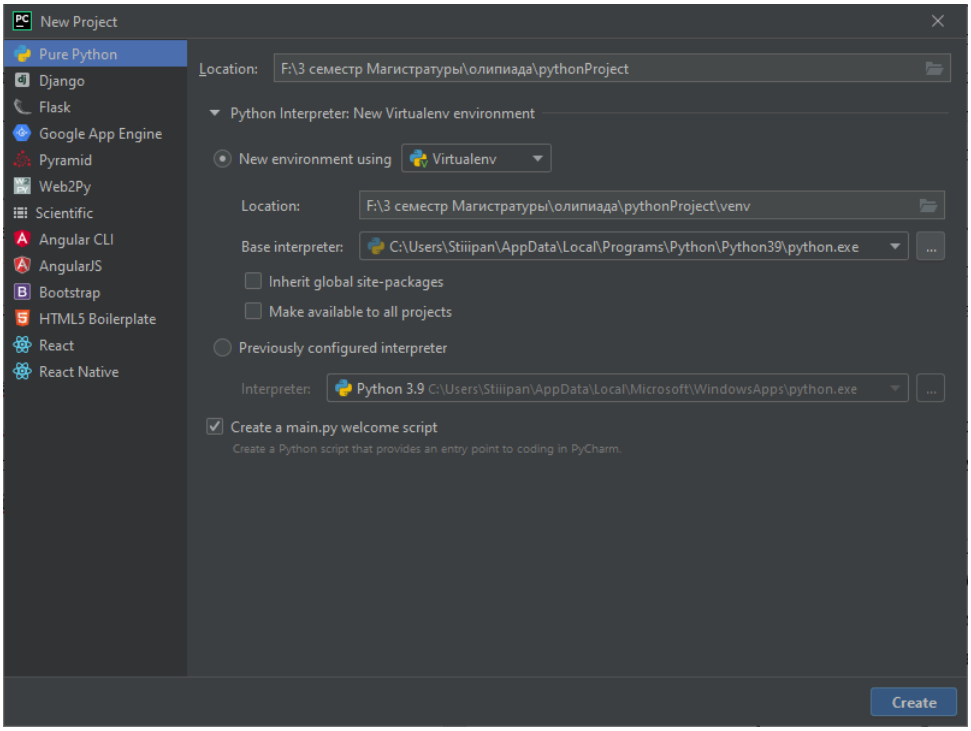

Рисунок 15 – Этап создания нового проекта

2.4 Обоснование выбора программно-технического обеспечения

2.4.1 Выбор языка программирования

В качестве языка программирования выступает Python, версии выше 3.8.

Питон имеет огромное количество библиотек и фреймворков, которые упрощает кодирование. А это значительно экономит время.

Самой популярной библиотекой используемой, для научных вычислений является NumPy, для более сложных вычислений – SciPy и для анализа данных scikit.

Эти библиотека работают вместе с мощными платформами, такими как TensorFlow, CNTK и Apache Spark. Эти библиотеки и фреймворки становятся

неотъемлемой частью проекта, когда речь заходит о проектах машинного обучения и глубокого обучения [3].

Простота.

Код Python лаконичен и удобочитаем даже для новых разработчиков, что выгодно для проектов по машинному и глубокому обучению. Благодаря простому синтаксису разработка приложения на Python выполняется быстрее, чем на многих языках программирования. Кроме того, это позволяет разработчику тестировать алгоритмы без их реализации.

Читаемый код также жизненно важен для совместного кодирования. Многие люди могут работать вместе над сложным проектом.

В команду можно легко найти Python-разработчика, так как Python – известная платформа. Таким образом, новый разработчик может быстро ознакомиться с концепциями Python и мгновенно приступить к работе над проектом.

Большое онлайн сообщество.

Python – это язык программирования с открытым исходным кодом, который пользуется отличной поддержкой со стороны многих ресурсов и качественной документации по всему миру. Он также имеет большое и активное сообщество разработчиков, которые оказывают помощь на любом этапе разработки.

Многие ученые приняли Python для проектов машинного и глубокого обучения, что означает, что большинство самых ярких умов мира можно найти в сообществах Python.

Быстрая разработка.

Python имеет простой для понимания и дружественный синтаксис. Кроме того, многочисленные фреймворки и библиотеки ускоряют разработку программного обеспечения. Используя готовые решения, многое можно сделать с помощью нескольких строк кода. Python хорош для разработки прототипов, что повышает производительность.

Гибкая интеграция.

Проекты Python можно интегрировать с другими системами, написанными на разных языках программирования. Это означает, что его гораздо проще смешать с другими проектами ИИ, написанными на других языках.

Кроме того, поскольку Python является расширяемым и переносимым, его можно использовать для выполнения межъязыковых задач. Адаптивность Python позволяет ученым и разработчикам легко обучать модели машинного обучения.

Быстрые тесты кода.

Python предоставляет множество инструментов для проверки кода и тестирования. Разработчики могут быстро проверить правильность и качество кода.

Проекты ИИ, как правило, занимают много времени, поэтому необходима хорошо структурированная среда для тестирования и проверки на наличие ошибок. Python - идеальный язык, поскольку он поддерживает эти функции.

Производительность.

Некоторые разработчики утверждают, что Python относительно медленный по сравнению с другими языками программирования. Несмотря на то, что скорость не является одной из сильных сторон Python, она предоставляет решение, известное как Cython. Это расширенный набор языка Python, предназначенный для достижения производительности кода, такой же, как язык C.

Разработчики могут использовать Cython для кодирования расширений C так же, как они пишут код на Python, поскольку его синтаксис почти такой же. Cython значительно увеличивает производительность языка.

Инструменты визуализации.

Python поставляется с широким набором библиотек. Некоторые из этих фреймворков предлагают хорошие инструменты визуализации. В ИИ, машинном обучении и глубоком обучении важно представлять данные в удобочитаемом формате. Поэтому Python - идеальный выбор для реализации этой функции.

Некоторые библиотеки, такие как Matplotlib, позволяют специалистам по данным создавать диаграммы, гистограммы и графики для лучшего представления данных и визуализации. Кроме того, различные API, поддерживаемые Python, улучшают процесс визуализации.

2.4.2 Выбор базы данных

MongoDB - документоориентированная система управления базами данных с открытым исходным кодом. Для хранения данных используется JSON-подобный формат. Эта СУБД отличается высокой доступностью, масштабируемостью и безопасностью.

Главные особенности MongoDB:

– Это кроссплатформенная документоориентированная база данных NoSQL с открытым исходным кодом.

– Она не требует описания схемы таблиц, как в реляционных БД. Данные хранятся в виде коллекций и документов.

– Между коллекциями нет сложных соединений типа JOIN, как между таблицами реляционных БД. Обычно соединение производится при сохранении данных путем объединения документов.

– Данные хранятся в формате BSON (бинарные JSON-подобные документы).

– У коллекций не обязательно должна быть схожая структура. У одного документа может быть один набор полей, в то время как у другого документа - совершенно другой (как тип, так и количество полей).

В одном документе могут быть поля разных типов данных, данные не нужно приводить к одному типу. Основное преимущество MongoDB заключается в том, что она может хранить любые данные, но эти данные должны быть в формате JSON.

СУБД MongoDB полагается на концепции базы данных, коллекций и документов. Для лучшего понимания, они ниже сравнены с терминами из языка структурированных запросов (SQL):

База данных - это физический контейнер для коллекций.

Коллекция - группа документов MongoDB. В терминологии SQL это соответствует таблице.

Документ - запись в коллекции MongoDB, набор пар ключ-значение. В терминологии SQL это похоже на строку в таблице базы данных.

Поле - ключ в документе. В терминологии SQL похоже на столбец в таблице.

Встроенный документ - в терминологии SQL похоже на создание связей между несколькими таблицами, по которым разбросаны данные, что делается операциями JOIN.

Преимущества этой СУБД:

– Документоориентированная база - сохранение данных в формате документов вместо формата реляционного типа, это делает MongoDB очень гибкой и адаптируемой к бизнес-требованиям. Возможность хранения разных типов данных особенно важна при работе с большими данными, которые собираются из разных источников и не ложатся в одну структуру.

– Специальные запросы - MongoDB поддерживает поиск по полям, диапазонные запросы и поиск по регулярным выражениям. Могут быть сделаны запросы для возврата определенных полей в документах.

– Индексация - можно создать индексы для улучшения производительности поиска в MongoDB. Любое поле в документе может быть проиндексировано. Это обеспечивает высокую скорость работы СУБД.

– Репликация - эта СУБД может обеспечить высокую доступность с помощью наборов реплик. Набор реплик состоит из двух или более экземпляров MongoDB. Каждая реплика набора может выступать в роли первичной или вторичной. Первичная реплика - главный сервер, который взаимодействует с клиентом и выполняет все операции чтения/записи. Вторичные реплики сохраняют копию данных первичной реплики с помощью встроенной реплика-

ции. Если с первичной репликой что-то случилось, происходит автоматическое переключение на вторичную реплику, затем она становится основным сервером.

– Балансировка нагрузки - MongoDB использует концепцию шардинга для горизонтального масштабирования с помощью разделения данных между несколькими экземплярами БД. Она может работать на нескольких серверах, балансируя нагрузку и/или дублируя данные, чтобы поддерживать работоспособность системы в случае аппаратного сбоя.

– Возможность развернуть в облаке - вы получаете готовую к работе, оптимально сконфигурированную, масштабируемую и управляемую базу данных по запросу за две минуты.

– Доступность - MongoDB поддерживает все популярные языки программирования, ее можно использовать бесплатно как open source решение.

Недостатки MongoDB:

– Эта база данных не настолько соответствует требованиям ACID (атомарность, согласованность, изолированность и устойчивость), как реляционные базы данных.

– Транзакции с использованием MongoDB являются сложными

– В MongoDB нет положений о хранимых процедурах или функциях, поэтому не получится реализовать какую-либо бизнес-логику на уровне базы данных, что можно сделать в реляционных БД.

Базы данных NoSQL, такие как MongoDB, являются хорошим выбором, когда ваши данные ориентированы на документы и плохо вписываются в схему реляционной базы данных, когда вам нужно приспособиться к большому масштабу, когда вы быстро создаете прототипы и в некоторых других случаях использования [4].

3 ОСНОВНЫЕ ЭТАПЫ ПРАКТИЧЕСКОЙ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

Разработку программного продукта можно разбить на несколько этапов такие как, проектирование базы данных, разработка алгоритмов для сбора данных, алгоритмов для обработки данных, алгоритмов для создания модели машинного обучения, создание пользовательского интерфейса, создание серверной части приложения. Дальше будут подробно описаны перечисленные выше этапы в подробном формате.

3.1Проектирование базы данных

В информационной системе необходимо сформировать набор коллекций для хранения собранной информации.

– Коллекция «Maps» хранит данные обо всех игровых локациях.

– Коллекция «Players» хранит данные обо всех игроках и их статистику.

– Коллекция «Prepare» хранит данные итоговые данные передаваемые в модель машинного обучения.

– Коллекция «Stats» хранит все собранные данные о прошедших матчах.

– Коллекция «Teams» содержит данные обо всех командах.

– Коллекция «hash» содержит данные о собранных данных, для предотвращения повторного попадания в базу данных.

После чего, нужно сформировать спецификации атрибутов каждой коллекции. Спецификация имеет вид таблицы, которая содержит 5 столбцов – наименование атрибута, описание атрибута, тип данных, диапазон значений, пример атрибута. В ниже представленных таблицах 2 – 7 представлены спецификации атрибутов коллекций.

Таблица 2 - Спецификация атрибутов коллекции «Maps»

Название атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример атрибута
_id	Набор символов, однозначно определяющий документ	Object ID	-	61a3003146b412368cb2b766
Название локации	Наименование игровой локации	String	-	Cache
Логотип локации	Ссылка на изображение в base64	String	-	aHR0cHM6Ly9zdGF0aWM0cHM6Ly9zdGF0aW0cHM6Ly9zdGF0aW=

Таблица 3 - Спецификация атрибутов коллекции «Players»

Название атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример атрибута
_id	Набор символов, однозначно определяющий документ	Object ID	-	61a3003146b412368cb2b766
ID игрока	Идентификатор игрока	Integer	>0	11940
Ник игрока	Игровое имя игрока	String	-	nealan
Текущая команда	Наименование текущей команды	String	-	K23
Rating	Рейтинг игрока среди других	Double	>0	1.33
DPR	Смертей за раунд	Double	>0	0.61
Kast	Процент раундов в которых игрок получил очки	Double	>0	74.9
Impact	Измерение влияния на игру	Double	>0	1.45
Adr	Средний урон за раунд	Double	>0	88.2
Kpr	Убийств за раунд	Double	>0	0.85

Таблица 4 - Спецификация атрибутов коллекции «Prepare»

Название атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример атрибута
_id	Набор символов, однозначно определяющий запись	Object ID	-	61a3003146b412368cb2b766
Победитель	Название команды победителя	String	-	No IdeA
Rating команды 1	Средний рейтинг первой команды	Double	>0	0.822
DPR команды 1	Средний показатель смертей за раунд первой команды	Double	>0	0.822
Kast команды 1	Средний процент раундов в которых игроки первой команды получил очки	Double	>0	0.822
Impact команды 1	Средний влияния на игру игроков первой команды	Double	>0	0.822
Adr команды 1	Средний урон за раунд первой команды	Double	>0	0.822
Kpr команды 1	Среднее количество убийств первой команды	Double	>0	0.822
Rating команды 2	Средний рейтинг второй команды	Double	>0	0.822
DPR команды 2	Средний показатель смертей за раунд второй команды	Double	>0	0.822

Продолжение таблицы 4

Kast команды 2	Средний процент раундов в которых игроки второй команды получил очки	Double	>0	0.822
Impact команды 2	Средний влияния на игру игроков второй команды	Double	>0	0.822
Adr команды 2	Средний урон за раунд второй команды	Double	>0	0.822
Kpr команды 2	Среднее количество убийств второй команды	Double	>0	0.822

Таблица 5 - Спецификация атрибутов коллекции «Stats»

Название атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример атрибута
_id	Набор символов, однозначно определяющий запись	Object ID	-	61a3003146b412368cb2b766
Победитель	Название команды победителя	String	-	No IdeA
Первая команда	Название первой команды	String	-	No IdeA
Состав первой команды	Список игроков первой команды	Object	-	{team1_p1:"Freeman", team1_p2:"HZ", …}
Вторая команда	Название второй команды	String	-	Red Wolf

Продолжение таблицы 5

Состав второй команды	Список игроков второй команды	Object	-	{team2_p1:" DeStRoYeR", team2_p2:" Lieu", …}
Локация	Название игровой локации	String	-	Cobblestone

Таблица 6 - Спецификация атрибутов коллекции «Teams»

Название атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример атрибута
_id	Набор символов, однозначно определяющий запись	Object ID	-	61a3003146b412368cb2b766
Название команды	Наименование команды	String	-	K23
Ссылка на команду	Ссылка на личную страницу команды	String	-	https://hltv.org/team/7244/k23
Логотип команды	Ссылка на логотип команды	String	-	https://img-cdn.hltv.org/teamlogo/t.png
Состав команды	Массив объектов участников команды содержащих ссылку на профиль игрока, никнейм, ссылка на фотографию	Array of objects	-	[{"url":"https://hltv.org/player/11940/nealan", "name":"neaLaN", "imgURI":"https://img-cdn.hltv.org" }, {…}, …]

Таблица 7 - Спецификация атрибутов коллекции «hash»

Название атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример атрибута
_id	Набор символов, однозначно определяющий запись	Object ID	-	61a3003146b412368cb2b766
Хэш	MD5 хэш обработанной строки таблицы	String	-	3b545831fd92e68851d28ac5ee981337
Дата записи	Дата и время записи в коллекцию	Date	>2019	08/04/2022 08:31:16

Если реляционные базы данных хранят строки, то MongoDB хранит документы. В отличие от строк документы могут хранить сложную по структуре информацию. Документ можно представить как хранилище ключей и значений.

Ключ представляет простую метку, с которым ассоциировано определенный кусок данных.

Однако при всех различиях есть одна особенность, которая сближает MongoDB и реляционные базы данных. В реляционных СУБД встречается такое понятие как первичный ключ. Это понятие описывает некий столбец, который имеет уникальные значения. В MongoDB для каждого документа имеется уникальный идентификатор, который называется _id. И если явным образом не указать его значение, то MongoDB автоматически сгенерирует для него значение [5].

По составленным выше спецификациям было решено составить модель базы данных. Для более удобного представления создаваемой базы данных на рисунке, коллекции представлены в виде схожем с логическим этапом проектирования реляционных баз данных. Таким образом каждая коллекция содержит название, под которым располагается первичный ключ с маркером PK,

после которого перечислены остальные атрибуты. Кроме того, для удобство было принято решение показывать атрибуты вместе с типами данных. Полученная таким образом структура представлена на рисунке 16.

Рисунок 16 – Модель проектируемой базы данных

3.2 Разработка алгоритмов для сбора данных

В этом пункте будут описаны принципы разработки скраппера и синтаксического анализатора.

Одним из первых с чем необходимо определиться на текущем этапе, это, как и с помощью чего будет происходить запросы на веб-страницы без помощи браузера, это необходимо для работы скрипта скраппера на сервере, где как правило отсутствует визуальный интерфейс. Для этих целей следует выбрать библиотеку языка Python, которая позволит посылать запросы. Для этих целей выбрана библиотека – Requests, из-за неимения выбора.

Эта библиотека позволяет легко отсылать HTTP/1.1 запросы (рис. 17), позволяя тем самым взаимодействовать с веб-приложениями. Это необходимо для решения задачи, связанной с передачей информации от пользователя к серверу и обратно.

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
'{"type":"User"...'
>>> r.json()
{'private_gists': 419, 'total_private_repos': 77, ...}
```

Рисунок 17 – Пример работы библиотеки Requests

После того как разобрались с инструментом для посылки запросов, теперь необходимо как-то разобрать страницу, для этого выбрана библиотека Beautiful Soup 4. Итак, с помощью данной библиотеки становится возможным взаимодействие Python с содержимым данных файлов XML и HTML, последние из которых нам и подходят.

Перед созданием скраппера необходимо хотя бы немного изучить тематику объектной модели документа (DOM-дерево), для взаимодействия с элементами страницы. Минимальным требованием является умение разбираться в CSS селекторах, в работе понадобятся селекторы типов элементов, классов, родителей и атрибутов (рис. 18). Селектор определяет, к какому элементу применять то или иное CSS-правило.

```
a { color: red; }
ul { margin-left: 0; }


.error {
color: red;
}
```

Рисунок 18 – Пример селектора тэга (сверху) и класса (снизу)

Теперь вооружившись двумя библиотеками, можно приступать к созданию скраппера. Первым делом необходимо определиться какой элемент на странице нас удовлетворяет. Для этого нужно провести инспектирование страницы (рис. 19), чтобы определить какой селектор использовать для отбора элементов DOM-дерева страницы.

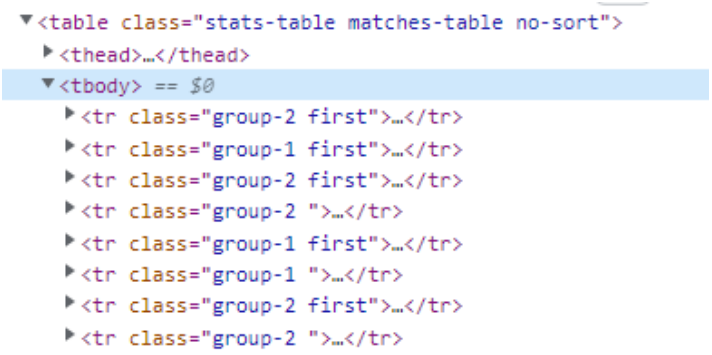

Рисунок 19 – Инспектирование страницы

Также необходимо определить граничные условия для скраппера, таким условием является отсутствие таблицы на странице и невозможность перейти на следующую страницу пагинации. Если с первым условием ситуация простая – проверка имеется ли на странице элемент tbody, являющийся дочерним для table с классами: stats-table, matches-table. А вторым условием является наличие на странице <a class=”pagination-next”>, с непустым атрибутом href, в котором содержится ссылка на следующую страницу с данными.

Скраппер собирает подходящий под условия tbody. Имея данные тэга, необходимо провести синтаксический разбор. Так как мы имеем конечное

число элементов в таблице, то наилучшим способом будет написать цикл, позволяющий перебрать их все.

Данные, попадаемые под сбор – Названия команд, локация, полученные очки командами и составы команд на момент встречи. Получить все кроме последнего не составляет труда, ведь эти данные имеются внутри текущей таблицы (рис. 20).

11/4/22	Totalne Gituwy (12)	Galaxy Racer fe (16)	Vertigo	ESL Impact Spring 2022 C...
11/4/22	Galaxy Racer fe (16)	Totalne Gituwy (7)	Dust2	ESL Impact Spring 2022 C...
11/4/22	ouroboros (8)	Nemiga (16)	Mirage	ESEA Spring 2022 Cash C...
11/4/22	ouroboros (16)	Nemiga (10)	Nuke	ESEA Spring 2022 Cash C...
11/4/22	Nemiga (16)	ouroboros (8)	Overpass	ESEA Spring 2022 Cash C...
11/4/22	VOID (4)	Oilers (16)	Mirage	Komplett Norway Cup 20...
11/4/22	Oilers (16)	VOID (2)	Vertigo	Komplett Norway Cup 20...

Рисунок 20 – Содержимое таблицы

Для получения состава команд, необходимо перейти на детальную страницу текущего матча. На детальной странице необходимо найти таблицы с данными, участвовавших команд (рис. 21).

Gaimin Gladiators	K (hs)	A (f)	D	KAST	K-D Diff	ADR	FK Diff	Rating 2.0
Bwills	21 (15)	7 (2)	11	90.0%	+10	118.3	+2	**1.76**
cynic	20 (7)	7 (1)	10	90.0%	+10	104.2	+3	**1.71**
nosraC	19 (11)	5 (3)	10	75.0%	+9	92.0	+1	**1.41**
jeorgesnorts	17 (8)	3 (1)	8	85.0%	+9	77.7	0	**1.32**
JazzPimp	14 (8)	5 (3)	9	85.0%	+5	73.6	0	**1.18**

Рисунок 21 – Отчет о прошедшем матче

Так как в разработанной системе имеется автоматический и автоматизированные алгоритмы по сбору и данных. При работе с автоматическим сборщиком, следует после сбора всех статистических данных их занести в базу данных. Для работы с документно-ориентированной базой данных MongoDB используется рекомендуемая разработчиками библиотека PyMongo. Помимо

средств для работы с БД, необходимо использовать библиотеку Dnspython, используемая для запросов, динамических обновлений и других вещей. Данные вносятся в конце каждой итерации цикла, таким образом, информация вносится по одному экземпляру за раз.

При работе с автоматизированным алгоритмом, по окончанию работы скраппера, все данные сохраняются в файл для их дальнейшей передачи в парсер. В связи с тем, что данные сохраняются в файл, необходимо чтобы функция реализующая парсинг одним из параметром принимала был относительный путь до этого файла, а вторым путь для выходного файла.

Перед началом синтаксического анализа, необходимо открыть файл, путь которого был передан в аргументах. После успешного открытия, необходимо считать файл в переменную, которую тут же передать в конструктор класса BeautifulSoup. Логика перебора данных здесь схожа с автоматическим парсером – в цикле перебираем все строки таблиц и считываем данные, «проваливаясь» в детальную статистку о матче. В конце каждой итерации данные пишутся в переменную, перед окончанием работы функции данные записываются в файл, расположенный по пути, который был передан в аргументах. Выполнив запись в файл, собранные данные необходимо записать в соответствующую коллекцию в базе данных.

Помимо сбора информации о пройденных встречах, необходимо иметь статистику об игроках. Для этих целей необходимо вновь провести скраппинг и дальнейший парсинг данных, но уже на страницах с немного другой структурой DOM-дерева. Статистика каждого пользователя понадобится в дальнейшем при подготовке данных.

Среди собираемых данных для каждого игрока, присутствуют такие, как идентификатор игрока на сайте с которого совершается скраппинг информации, никнейм, текущая команда и пред посчитанные данные: raiting – говорящий о том как игрок показывает себя в сравнении с другими игроками, dpr – смерти за раунд, kast – процент раундов в которых игрок совершил или полу-

чил помощь в поражении вражеского игрока, impact – величина показывающая, сколько помощи делает тот или иной человек, adr – средний нанесенный урон за раунд, kpr – среднее количество пораженных игроков (рис. 22).

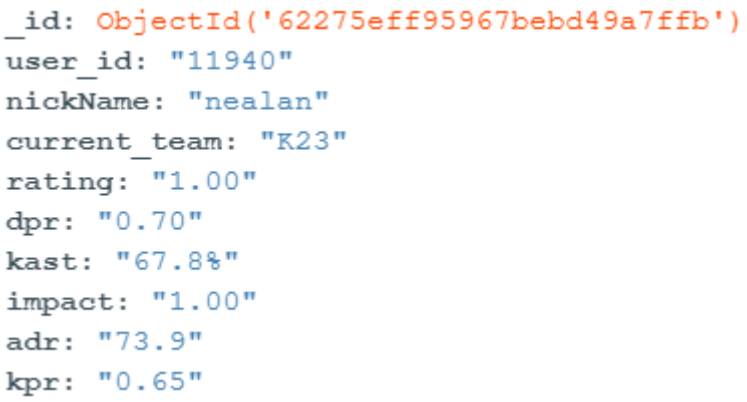

Рисунок 22 – Данные собираемые об игроках

Также необходимо собрать данные о командах, которые участвуют в соревновательных играх. Среди всех команд, будем выбирать лишь те, у которых имеется текущий игровой состав. Таким образом, команды у которых нет утвержденной команды, не будут попадать в выборку, и тем самым статистические данные с такими командами не будут участвовать в дальнейшей работе. Эти данные необходимы будут для построения пользовательского интерфейса.

В командах нас интересуют такие данные как, название команды, ссылка на логотип команды, ссылка на страницу профиля команды и состав команды – 5 игроков, для которых необходимо получить их никнеймы, ссылки на их профили, фотографии игроков (рис. 23).

Рисунок 23 – Данные о командах

3.3 Разработка алгоритмов для обработки данных

На данном этапе из ранее перечисленных библиотек используется лишь модули для работы с документно-ориентированной системой управления базами данных MongoDB, а именно реализация для языка Python – pymongo.

Так как для большинства алгоритмов машинного обучения все данные необходимо перевести в числовое значение, то на этом этапе необходимо заняться их подготовкой.

Для обработки данных необходимо перемножить соответствующие значения из двух таблиц и записать результаты в другую таблицу, это необходимо для дальнейших шагов в разработке приложения. Первым делом необходимо создать курсоры на коллекции со статистикой и игроками. И так как нас больше интересует собранная статистика, нежели сами игроки, то перебор будет осуществляться именно по первой коллекции.

Процесс подготовки данных заключается в вычислении средних показателей игроков по команде, таких как rating, dpr, kast, impact, adr и kpr. Для этих

целей и понадобятся собранные ранее данные по игрокам. В начале каждой итерации необходимо получать количество игроков участвующих в команде, которую сейчас обрабатываем. Чтобы получить вышеперечисленные показатели, необходимо обращаться к курсору с командами, но ограничиться лишь теми игроками, у которых в поле текущей команды стоит, команда из данной итерации. Получив курсор с игроками, перебираем их и складываем соответствующие значения в переменные приводя их к значению числа с плавающей точкой.

Кроме того, необходимо определить победителя, для этих целей переопределим это в бинарный формат, то есть если победителем была первая команда, в результат попадет значение 0, в противном случае 1. Записываем это в базу данных вставляя по одному документу в конце каждой итерации. В результате работы, получается коллекция, состоящая из документов как показано на рисунке 24.

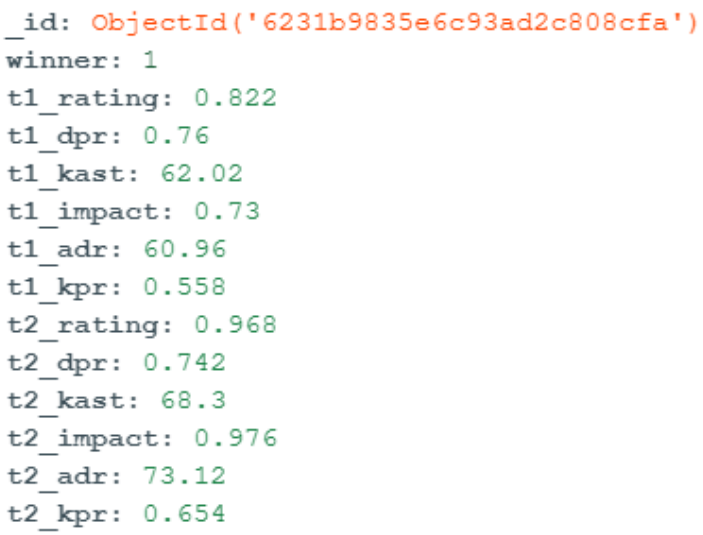

Рисунок 24 – Результат подготовки данных

3.4 Разработка алгоритмов для создания модели машинного обучения

Приступая к самой значимой части информационной системы – к моделям машинного обучения, нельзя не отметить их многообразие (рис. 25). Ведь, верно, выбранный метод поможет лучше справиться с поставленной задачей.

Рисунок 25 – Виды машинного обучения

Как можно видеть на рисунке выше машинное обучения включает в себя различные направления, которые также делятся на подкатегории, которые в свою очередь состоят из определенных направлений.

3.3.1 Классическое обучения

При использовании алгоритмов машинного обучения без учителя машина сама должна найти среди хаотичных данных верное решение и отсортировать объекты по неизвестным признакам. Например, определить, где на фото собака.

Эта модель применяется для данных, которые просто невозможно разметить из-за их колоссального объема. Такие алгоритмы применяют для риск-менеджмента, сжатия изображений, объединения близких точек на карте, сегментации рынка и других.

Если же используется другая категория алгоритмов, так называемые обучение с учителем – когда у машины есть некий учитель, который знает, какой ответ правильный. Это значит, что исходные данные уже размечены (отсортированы) нужным образом, и машине остается лишь определить объект с нужным признаком или вычислить результат.

Такие модели используют в спам-фильтрах, распознавании языков и рукописного текста, выявлении мошеннических операций, расчете финансовых показателей, скоринге при выдаче кредита. В медицинской диагностике классификация помогает выявлять аномалии – то есть возможные признаки заболеваний на снимках пациентов.

3.3.2 Обучение с подкреплением

Это более сложный вид обучения, где искусственному интеллекту нужно не просто анализировать данные, а действовать самостоятельно в реальной среде – будь то улица, дом или видеоигра. Задача робота – свести ошибки к минимуму, за что он получает возможность продолжать работу без препятствий и сбоев.

Обучение с подкреплением инженеры используют для беспилотников, роботов-пылесосов, торговли на фондовом рынке, управления ресурсами компании. Именно так алгоритму AlphaGo удалось обыграть чемпиона по игре Го: просчитать все возможные комбинации, как в шахматах, здесь было невозможно.

3.3.3 Ансамблевые методы

Это группы алгоритмов, которые используют сразу несколько методов машинного обучения и исправляют ошибки друг друга. Их получают тремя способами:

– Стекинг – когда разные алгоритмы обучают по отдельности, а потом передают их результаты на вход последнему, который и принимает решение;

– Беггинг – когда один алгоритм многократно обучают на случайных выборках, а потом усредняют ответы;

– Бустинг – когда алгоритмы обучают последовательно, при этом каждый обращает особое внимание на ошибки предыдущего.

Ансамбли работают в поисковых системах, компьютерном зрении, распознавании лиц и других объектов.

3.3.4 Нейросети и глубокое обучение

Самый сложный уровень обучения ИИ. Нейросети моделируют работу человеческого мозга, который состоит из нейронов, постоянно формирующих между собой новые связи. Очень условно можно определить их как сеть со множеством входов и одним выходом. Нейроны образуют слои, через которые последовательно проходит сигнал. Все это соединено нейронными связями – каналами, по которым передаются данные. У каждого канала свой «вес» – параметр, который влияет на данные, которые он передает.

Искусственный интеллект собирает данные со всех входов, оценивая их вес по заданным параметрами, затем выполняет нужное действие и выдает результат. Сначала он получается случайным, но затем через множество циклов становится все более точным. Хорошо обученная нейросеть работает, как обычный алгоритм или точнее.

Настоящим прорывом в этой области стало глубокое обучение, которое обучает нейросети на нескольких уровнях абстракций.

Здесь используют две главных архитектуры:

– Сверточные нейросети первыми научились распознавать неразмеченные изображения – самые сложные объекты для ИИ. Для этого они разбивают их на блоки, определяют в каждом доминирующие линии и сравнивают с другими изображениями нужного объекта;

– Рекуррентные нейросети отвечают за распознавание текста и речи. Они выявляют в них последовательности и связывают каждую единицу – букву или звук – с остальными.

Нейросети с глубоким обучением требуют огромных массивов данных и технических ресурсов. Именно они лежат в основе машинного перевода, чатботов и голосовых помощников, создают музыку и дипфейки, обрабатывают фото и видео [6].

3.3.5 Разработка решения поставленной задачи

Из имеющихся категорий алгоритмов для машинного обучения, для решения имеющейся задачи подходят алгоритмы классического обучения, а именно обучение с учителем. Учителем в данном случае будут выступать ранее собранные и подготовленные данные.

Для работы с машинным обучением в языке программирования Python имеется ряд библиотек, пригодных для этой работы. Среди таких библиотек можно выделить TensorFlow, Keras, SciPy и Scikit-learn.

TensorFlow — это одна из самых популярных Python библиотек для создания нейронных сетей. Она использует многомерные массивы, также известные как тензоры, которые позволяют производить несколько операций на одних входных данных. Благодаря своей многопоточной природе, он может одновременно тренировать несколько нейронных сетей и создавать высокоэффективные и масштабируемые модели.

Keras в основном применяют для создания моделей глубокого обучения и нейронных сетей. Он использует TensorFlow и Theano и позволяет легко создавать нейронные сети. Из-за того, что Keras генерирует вычислительный граф на сервере, он немного медленнее других библиотек.

Отличительная черта библиотеки SciPy — функции, которые полезны в математике и других науках. Например, статистические функции, функции оптимизации, обработки сигналов. Для решения дифференциальных уравнений и оптимизации он включает в себя функции для нахождения численного решения интегралов. Важные сферы его применения:

– многомерная обработка изображений;

– решения преобразований Фурье и дифф. уравнений;

– благодаря оптимизированным алгоритмам, он может выполнять вычисления линейной алгебры очень эффективно и с высокой надёжностью.

Scikit-learn, это самая важная библиотека для машинного обучения на Python. После очистки и манипуляций с данными в Pandas или NumPy, Scikit-learn используется для создания моделей машинного обучения. Библиотека предоставляет множество инструментов для предиктивного моделирования и анализа.

Также незаменимым для работы с большими объемами данных является использование набора из двух инструментов, таких как Pandas и NumPy.

Первый используется для анализа данных, являющийся одним из самых популярных инструментов. Он предоставляет множество полезных инструментов для сбора, очистки и моделирования данных. С Pandas появляется возможность загружать, подготавливать, анализировать и манипулировать любыми индексированными данными. Библиотеки для машинного обучения также используют датафреймы из Pandas в качестве входных данных.

NumPy, имеет большое преимущество в своей работе – поддержка n-размерных массивов. Эти многомерные массивы в 50 раз надёжнее списков в Python. Благодаря им, NumPy очень популярна в анализе данных.

NumPy часто используют другие библиотеки типа TensorFlow, для внутренних вычислений с тензорами. Библиотека предлагает быстрые универсальные функции для рутинных вычислений, которые сложно делать вручную.

NumPy использует функции оптимизированные для работы с многомерными массивами, скорость работы которых сравнима с MATLAB.

Так как данные хранятся в удаленной базе данных, перед началом работы с алгоритмами машинного обучения необходимо получить все заранее подготовленные данные из коллекции. Для чтения из базы все также используется библиотека Pymongo. Для удобства разработки, чтения и редактирования код следует разбить на функции.

Функция для получения данных из базы будет возвращать список из данных хранящихся в коллекции. При запуске скрипта первым делом выполняется подключение к базе данных, тем самым создается переменная с глобальным контекстом выполнения, доступная в том числе и в этой функции. Создание такой переменной необходимо, для того чтобы избежать многочисленных подключений к системе управления базой данных в пределах одного скрипта. Итак, создав переменную теперь есть возможность создать курсор на нужную коллекцию. Теперь единственное что осталось – провести перебор по курсору, записывая данные в переменную, которую в конце функции необходимо будет передать на выход.

Функцию для получения данных необходимо вызвать сразу после подключения к базе данных, ведь дальнейшая работа напрямую зависит от полученных данных, значение функции необходимо записать в переменную для удобства разработки. После завершения работы функции необходимо создать объект, ранее рассмотренной библиотеки Pandas, а именно объект класса DataFrame который позволяет хранить данные в табличной форме. В конструктор класса, необходимо передать считанные из базы данных данные и определить заголовки столбцов, передав массив из строк в именованный параметр.

Создав объект с данными, необходимо определить данные которые непосредственно пойдут на обучение, и те данные на которые будут ориентироваться модель.

После необходимо привести все категориальные данные при их наличии в численные. Проведя данные манипуляции нужно разбить набор данных на

учебные и тестовые выборки, при этом перемешивая их, для получения наилучшего результата в процессе обучения.

После всех подготовительных процессов можно приступать к главной задаче реализуемого модуля, а именно создание модели машинного обучения. Для этих целей необходимо создать объект выбранного алгоритма, задать ему необходимые стартовые параметры при их наличии. Для решения поставленной задачи выбран алгоритм регрессионного градиентного бустинга, которому заданы параметры для обучения: количество стадий обучения, параметр отвечающий за случайный порядок данных на итерациях.

После создания классификатора модели, необходимо приступать к обучению модели, для этого используется созданные чуть ранее выборки для обучения. После успешного обучения необходимо провести тестирование, также на ранее созданных выборках. Время обучения и тестирования модели напрямую зависит от вычислительных мощностей компьютера, на котором производится обучение так и от размеров данных, передаваемых в модель (рис. 26).

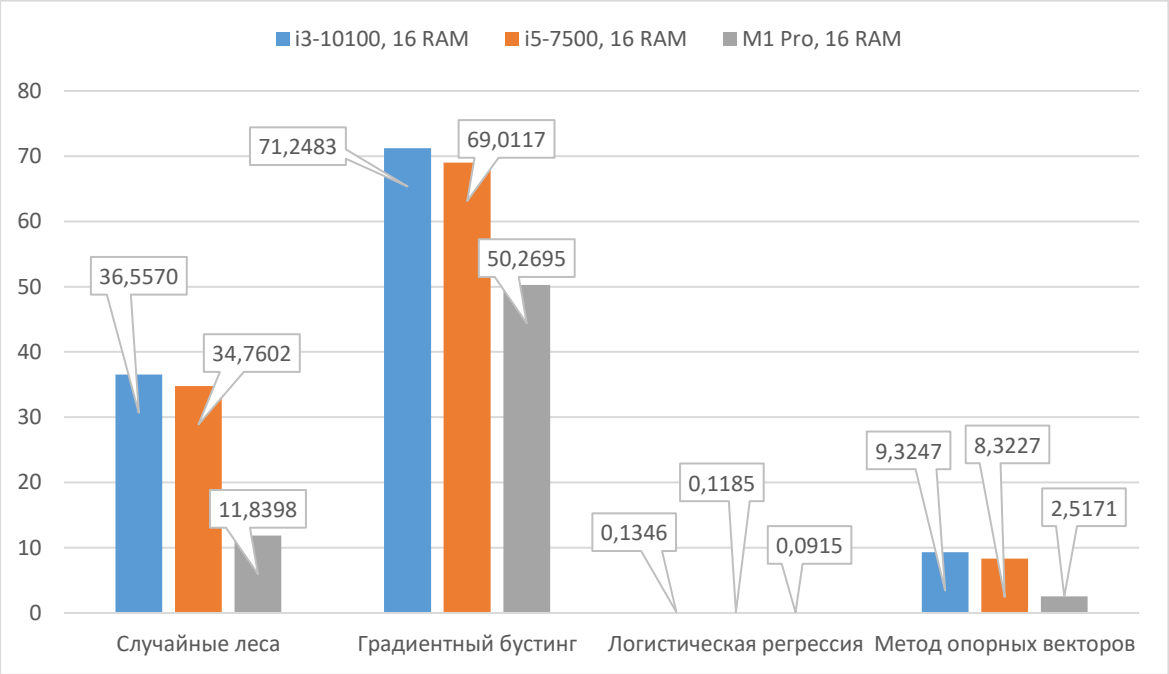

Рисунок 26 – График зависимости времени обучения от вычислительных мощностей

И для того, чтобы пользователи информационной системы получали наилучший опыт использования, после успешного проведения обучения и тестирования модель необходимо сохранить, и в дальнейшем использовать при необходимости.

3.5 Разработка серверной части

Так как разрабатываемая информационная система представлена в виде веб-сайта с динамической информацией, то это означает что у нее есть серверная часть, позволяющая обрабатывать запросы от пользователя, будь то отображение различных веб-страниц, обработка форм, динамическое изменение информации и другие.

В последние годы высокоуровневый язык программирования Python начал приобретать популярность в разработке веб-приложений, это стало возможным благодаря развивающимся фреймворкам.

Для решения поставленной задачи был выбран свободно распространяющийся Flask. Данная библиотека относится к микрофреймворкам. Он не ограничивает разработчиков в решениях, например, какой базой данных пользоваться. Единственное что она решает – какой шаблонизатор использовать, и это легко изменить. По умолчанию Flask поставляется вместе с шаблонизатором Jinja. Он позволяет формировать страницы на стороне сервера, тем самым снижая нагрузку на устройства пользователя.

Шаблонизатор представляет из себя программное обеспечение, которое позволяет использовать html шаблоны для получения результирующих страниц.

Приступая к написанию серверной части проекта, следует после подключения необходимых для работы библиотек, создать объект класса Flask, именно этот объект и будет решать все поставленные задачи. Для того чтобы сервер понимал какие страницы необходимо обрабатывать нужно создать набор правил, по которым и будет происходить определение для какой страницы то или иное правило необходимо применять. Также не стоит забывать про HTTP – методы, по которым тоже определяется состояние страницы. В

разработанной веб-приложении используются стандартные методы – GET и POST, позволяющие передавать значения на сервер.

Внутри этих правил и описываются основные действия сервера. Например, так как основная работа в веб-приложении заключается в действиях в пределах одной страницы, то здесь необходимо проверять в рамках какого метода сейчас происходит обработка ответа. Если передан запрос с методом GET, то сервер передает шаблонизатору необходимую страницу и данные, которые нужно разместить на странице. Среди передаваемых на клиентскую часть данных присутствует, список команд, содержащий логотип, название и описание каждого игрока, а также список игровых локаций с названием и логотипом.

В случае если методом запроса является POST, то сервер обработает тот же адрес, но уже по-другому. Первым делом необходимо получить переданные от клиентской части в теле запроса данные, коими являются названия команд, и локация на которой будет происходить их встреча. Получения этих данных становится возможным, благодаря разбору тела запроса на составные части, частью которого они и являются.

Так как все данные, собранные по игрокам, являются средними по всем игровым локациям, участвующими в текущем пуле карт, то перед началом предсказанием необходимо получить данные по конкретной карте для каждого игрока из команд. Для этого обратимся к соответствующей таблице в базе данных.

Получив данные их следует передать в предварительно подготовленную обученную модель для получения результата. В итоге с сервера в данном случае будет передаваться чуть больше информации нежели, при рендеринге странице с GET – запросом. К тем данным добавится ряд новых: данные, введенные пользователем, а также статистическая информация о проведенных и выигранных матчах каждой из команд на каждой из игровых локаций.

Для получения последних из базы данных необходимо воспользоваться более сложными запросами чем обычно, для этих целей в документно-ориентированной системе управления базами данных MongoDB имеется метод для

агрегирования данных. Данный метод позволяет проводить сложные запросы к базе данных и манипулировать сгруппированными данными.

Для удобства разработки действия, не относящиеся к определению введенного пути и рендерингу информации для той или иной страницы, вынесены в отдельные функции. Именно так сделано для функций работающих с базой данных и с моделью машинного обучения.

3.6 Разработка клиентской части

Для разработки клиентской части информационной системы выбраны основные технологии для написания веб-страниц. Среди таких технологий язык разметки гипертекста пятой версии HTML5, каскадные таблицы стилей CSS3, а также объектно-ориентированный язык JavaScript.

Для верстки страниц используется язык разметки HTML. С его помощью создается необходимая для проекта структура страниц. Сверстанные страницы должны быть валидными, то есть соответствовать критериям, которые утвердила The World Wide Web Consortium. Среди критериев больше всего интересуют кроссбраузерность и высокую скорость загрузки.

Кроссбраузерность говорит о совместимости сайта с популярными браузерами. То есть один и тот же сайт может выглядеть по-разному на различных устройствах и браузерах. Разработка велась в браузере Chrome, также проверка была проведена и в других популярных браузерах (рис. 27). Данный браузер был выбран основным на основе статистики.

И второй интересующий критерий – скорость загрузки. Этот критерий отвечает за отсутствие ошибок в коде страниц, так как они уменьшают скорость загрузки страниц, даже если не являются столь заметными [7].

В создание макета также участвуют данные из бэкэнд части, среди таких данных: списки команд, список игровых локаций. Для этих целей и используется библиотека Jinja, поставляемая вместе с Flask. С ее помощью можно встраивать циклы, условные операторы и просто выводить простые данные в разметке страницы, для этих целей используются наборы из символов: {, }, %. Различное их сочетание дает разные функциональные возможности. Данный

способ позволяет создавать страницы с динамической информацией, что позволяет не беспокоиться об изменении данных передаваемых на фронт-энд часть.

Создав макет страницы с использованием HTML нужно предать ему внешний вид, за это отвечают каскадные таблицы стилей CSS. CSS – технология описания внешнего вида документа.

Рисунок 27 -Популярные браузеры за 2020-2022 года

Для создания лучшего пользовательского опыта следует не забывать про адаптивную верстку. Данный подход в разработке сайтов предполагает отсутствие полос прокрутки и масштабируемых областей при просмотре на любом устройстве. Она позволяет менять дизайн страниц в зависимости от размера экрана, ориентации девайса, платформы и других. Этот подход является неотъемлемой частью в современной разработке веб-приложений. Например, на рисунках 28 и 29 показана одна и та же форма, но единственным отличием является различные разрешения экранов, на которых были сделаны скриншоты.

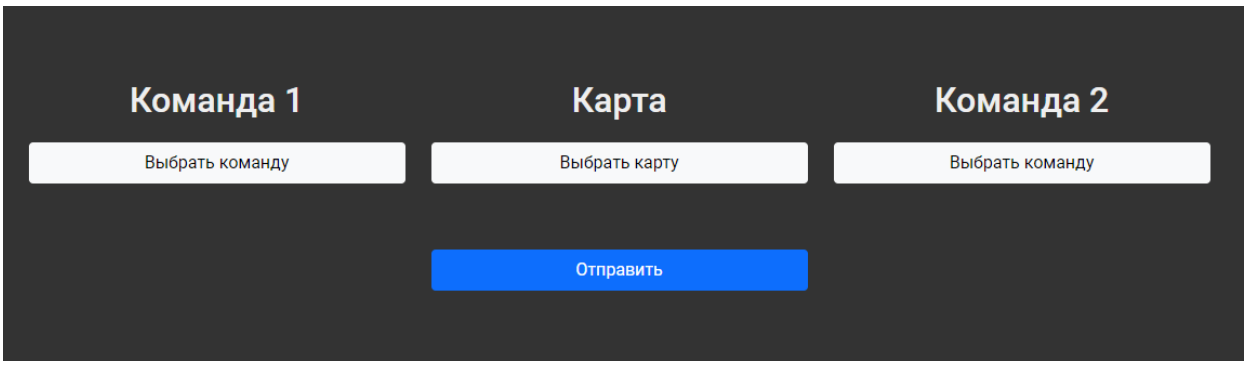

Рисунок 28 – Десктопная версия

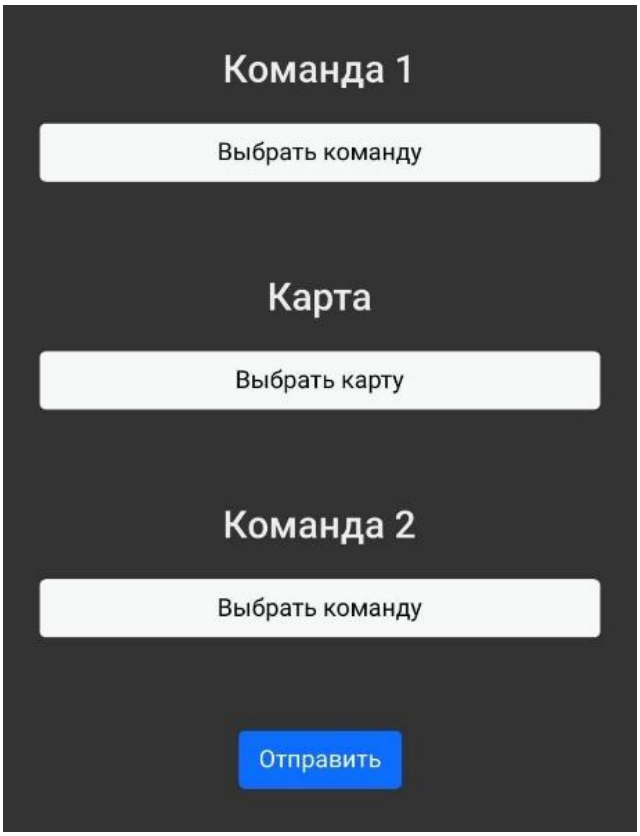

Рисунок 29 – Мобильная версия

Для отображения одних и тех же страниц используются правило @media, которые активируются только при определенных условиях, которыми в данном случае является размер экрана. В целях обеспечения качественной разработки отзывчивого приложения со стратегией mobile-first, выбрана одна из самых популярных фронт-энд инструментов, использующий отзывчивую сетку и JavaScript–расширения. Всеми этими требованиями обладает свободно распространяемый фреймворк Bootstrap.

Данный инструмент позволяет создавать элементы с готовым стилевым оформлением, и поведением. Например, все кнопки в разработанной информационной системе используют класс btn из этой библиотеки и также дополнены самописными стилями. По нажатию на кнопки выбора команд или локации происходит открытие соответствующего модального окна, предлагающего список с локациями или командами (рис. 30).

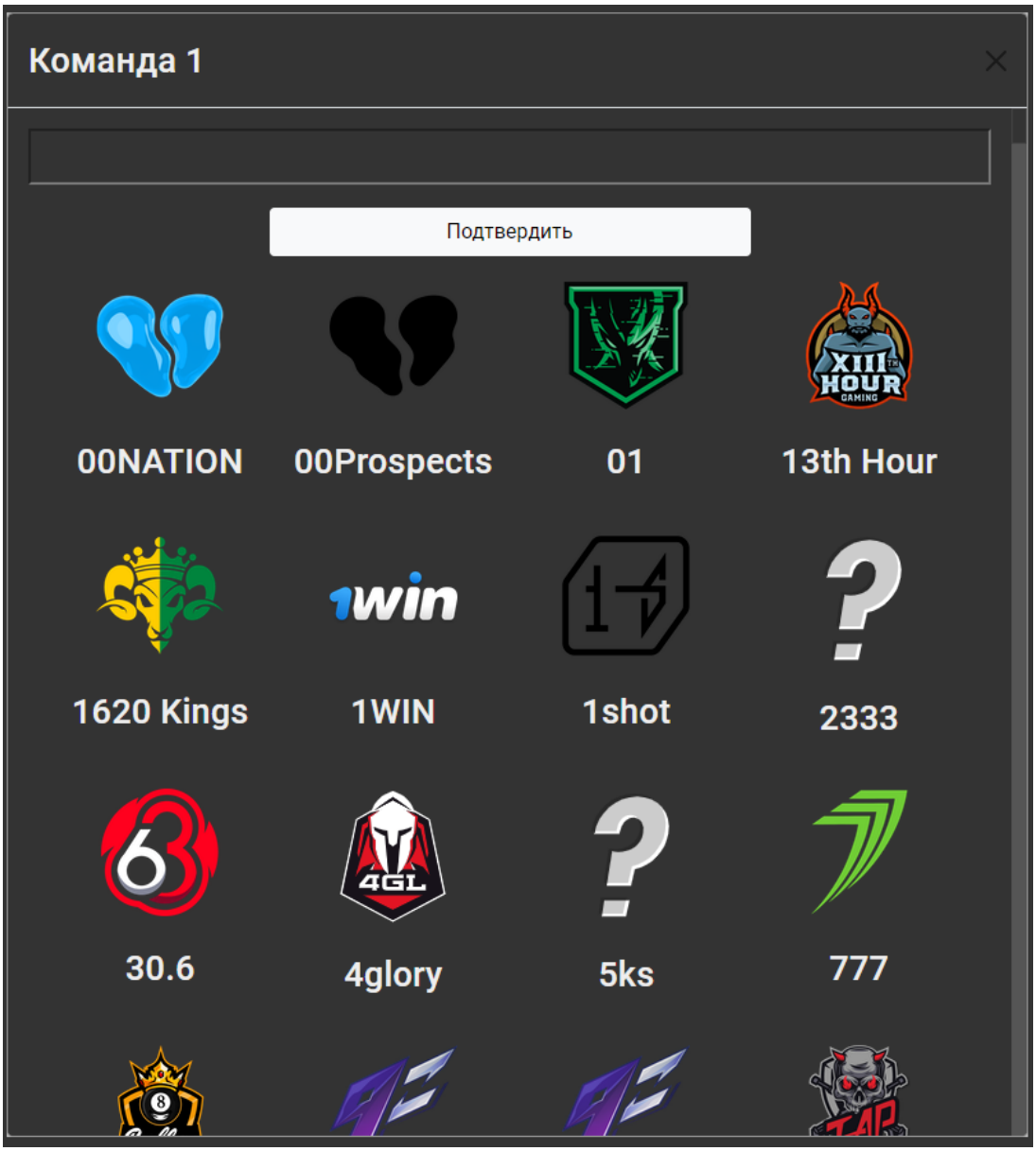

Рисунок 30 - Модальное окно выбора команды

JavaScript используется для написания скриптов, определяющих поведение тех или иных элементов на странице, он позволяет перенастроить их стандартные поведения или добавлять им его при их отсутствии. В помощь работы

с элементами DOM-дерева, используется библиотека JQuery, призванная уменьшить код и тем самым повысить скорость программирования.

Например, в модальном окне для удобства выбора команд предусмотрена возможность производить поиск по названию, если команда не соответствует вводимой строке, то она скрывается из области видимости пользователя. При выборе определенной команды «ячейка» с наименованием и логотипом подсвечивается, тем самым давая пользователю понять какая из команд выбрана, при повторном нажатии свечение будет убрано. При подтверждении команды, модальное окно скроется и на странице отобразятся логотип команды, а также ее состав с их фотографиями (рис. 31).

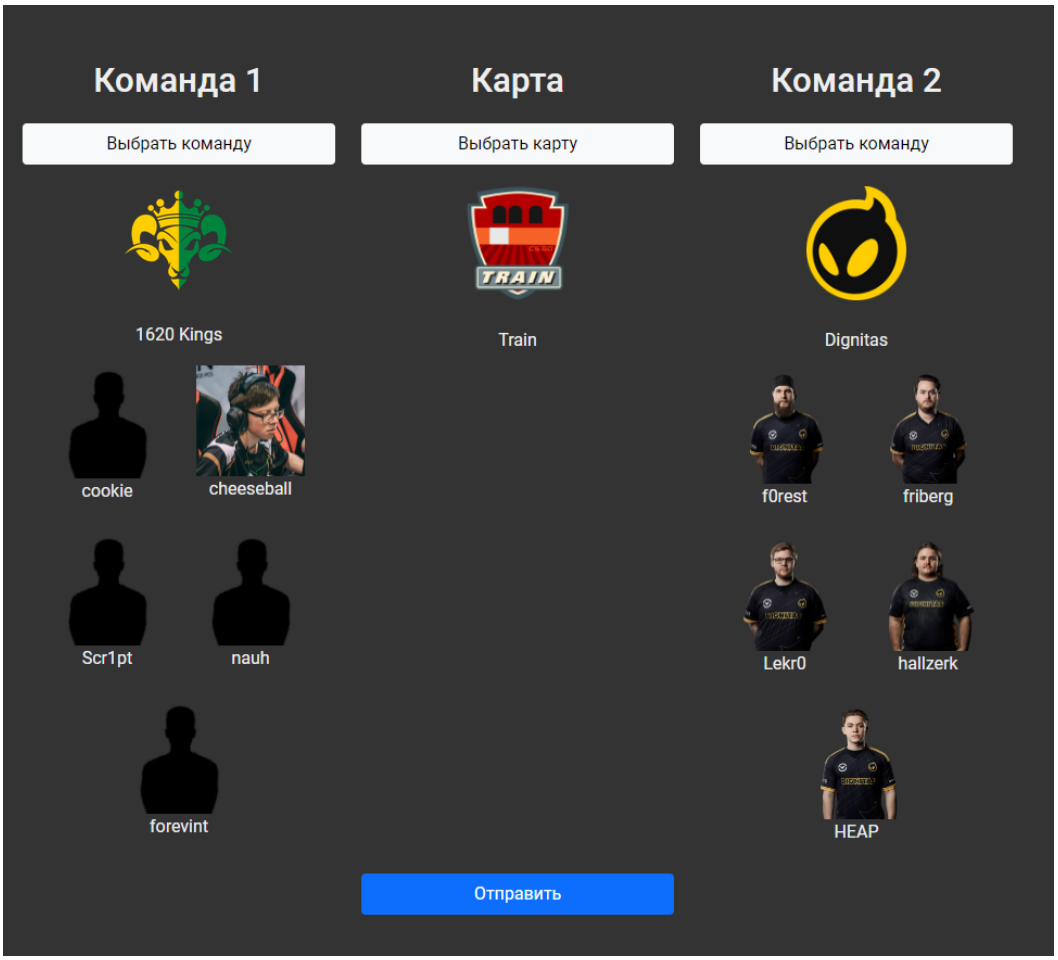

Рисунок 31 – Введенные данные

Так как информационная система, по сути, представляет 3 последовательных экрана, для удобства используется JavaScript библиотека для постраничной прокрутки fullpage.

Также помимо этих библиотек, имеется библиотека для визуализации данных. Среди множества популярных предлагаемых решений выбор был сделан на Chart.js, за ее гибкость настройки, легковесность и обширную документацию, что является неотъемлемым требованием при разработке. Пример представлен на рисунке 32.

В данном проекте эта библиотека позволяет строить сравнительных графики для каждой из команд, где по оси абсцисс перечисляются названия локаций, на которых команда когда-либо играла, а по оси ординат откладываются с динамическим шагом количество проведенных матчей.

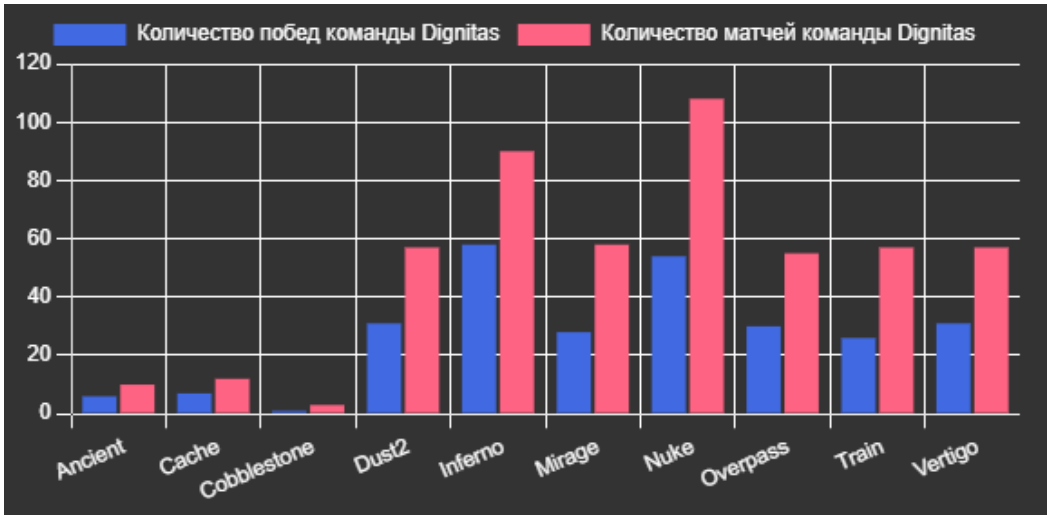

Рисунок 32 – Пример построенного графика

Также на клиентской части приложения предусмотрена проверка валидности вводимых данных. То есть, если пользователь введет не весь набор данных, то он будет оповещен соответствующим модальным окном. При вводе двух одинаковых команд, пользователь также будет оповещен.

3.7 Результаты фактического тестирования программного продукта

Для проверки работоспособности работы были выбраны профессиональные команды Natus Vincere в составе s1mple, electroNic, BoombI4, Perfecto и b1t, а также команда fnatic: KRIMZ, ALEX, Brollan, mezii и regali. В качестве локации был выбран Cache (рис. 33).

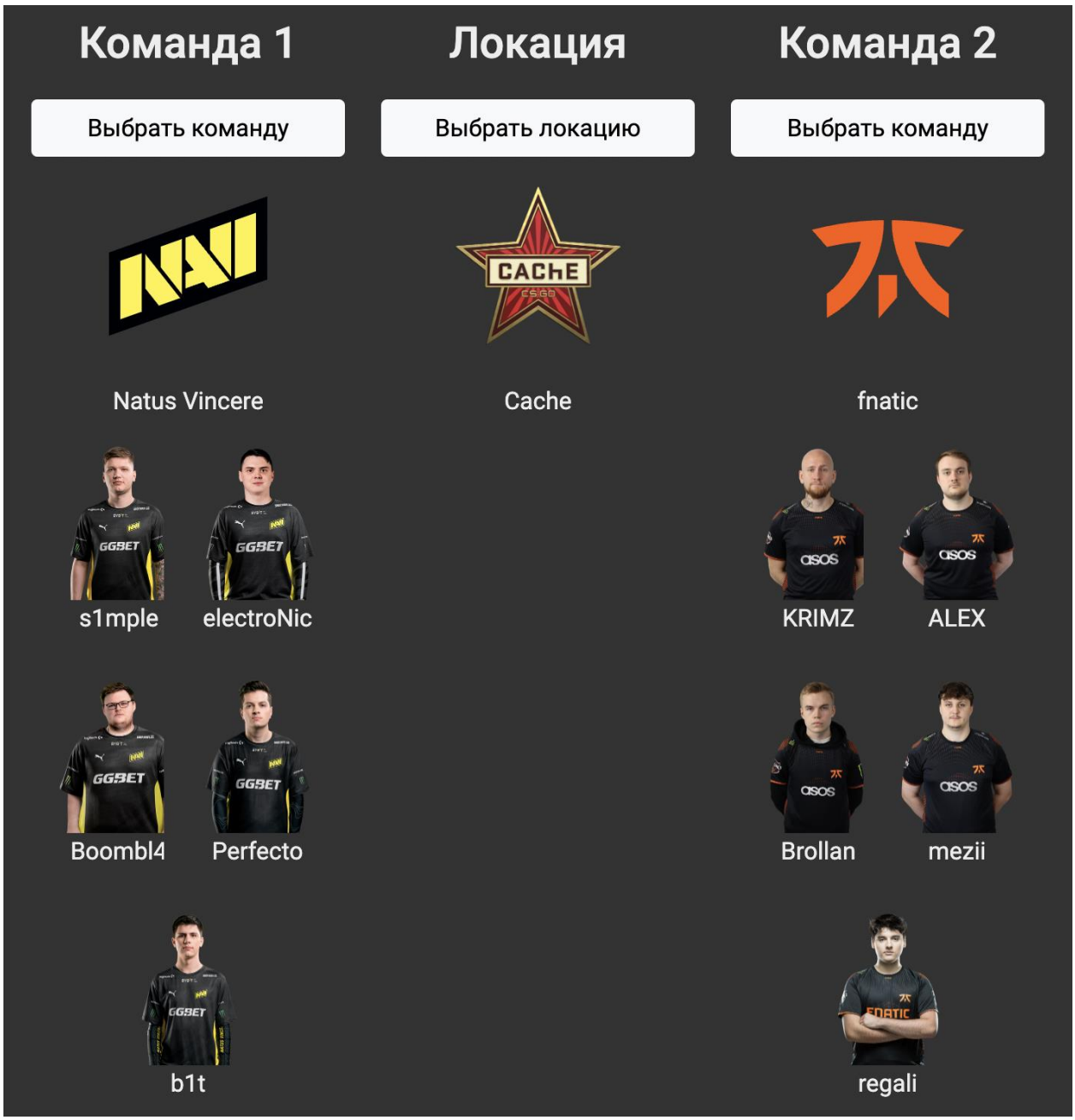

Рисунок 33 – Данные для тестирования

В результате обработки введенных данных, обученной ранее моделью машинного обучения, потенциальным победителем становится шведская команда fnatic. Что продемонстрировано на рисунке 34

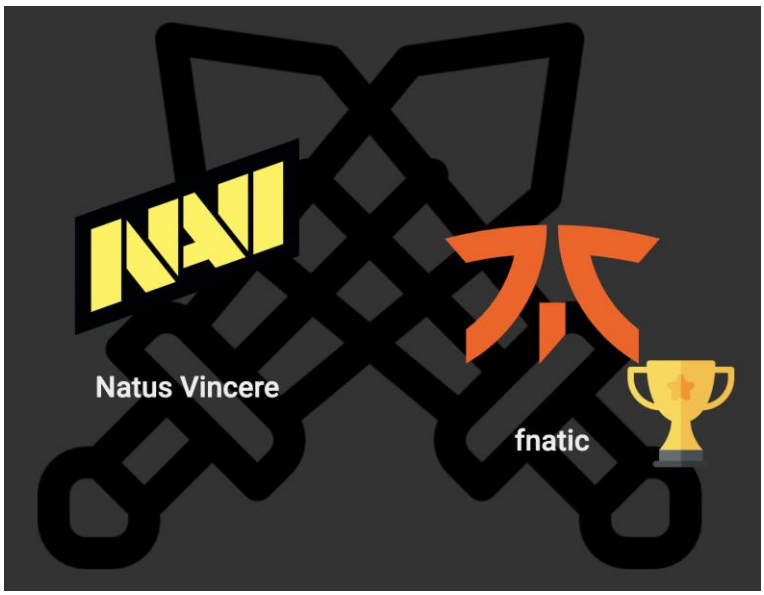

Рисунок 34 – Результат встречи

Также помимо результата, информационной системой были предоставлены графики, данные для которых были взяты напрямую из базы данных (рис. 35).

Рисунок 35 – Статистические данные команд

ЗАКЛЮЧЕНИЕ

В ходе выполнения научно – исследовательской работы была разработана информационная система, позволяющая сделать предсказание рейтинга команд в киберспорте используя технологии Большие данные. Разработка заключается в создании автономных алгоритмов по сбору и обработке данных с дальнейшим обучением модели машинного обучения, а также создания клиент-серверного приложения для взаимодействия с обученной моделью.

Для сбора данных используется связка технологий, позволяющая делать HTTP – запросы к сайту и синтаксический анализатор данных. Результат выполнения обрабатывается и записывается в нереляционную базу данных.

В качестве модели машинного обучения выступает модель градиентного бустинга, показавшая наилучшие результаты в процессе разработки и тестирования.

Для взаимодействия пользователя с системой предусмотрен веб-интерфейс, построенный на основе базовых технологий разработки веб - страниц. А в качестве серверной части используется микро-фреймворка Flask, отвечающий за обработку запросов со стороны браузера.

Охарактеризована предметная область проводимого исследования, в которой указаны сущность технологии больших данных и киберспорта. Проведен анализ существующих методов предсказания результатов в киберспорте.

Научная новизна работы определяется разработкой и реализацией способов взаимодействия различных технологий в программирование, по средствам создания правильно сконфигурированных и спрограммированных алгоритмов по сбору и обработке данных и обучению модели машинного обучения. А также взаимодействия пользователя с моделью машинного обучения через веб-интерфейс.

Практическую значимость будут иметь следующие результаты работы:

– разработка автоматических алгоритмов по сбору и обработке данных о результатах матчей, а также последующее обучение модели машинного обучения;

– взаимосвязь между пользовательским интерфейсом и серверной частью веб-приложения, а именно предсказание результатов матчей по введенным пользователем данных;

– динамическое построение веб-страницы основываясь на данных из нереляционной базы данных MongoDB.

БИБЛИОГРАФИЧЕСКИЕ ССЫЛКИ

1 Что такое большие данные? [Электронный ресурс]. – Режим доступа: https://www.oracle.com/ru/big-data/what-is-big-data/#use-cases – 02.02.2022

2 Киберспорт [Электронный ресурс]. – Режим доступа: https://znaki.fm/esports/ - 01.02.2022

3 Why Python is Good for Machine Learning [Электронный ресурс]. – Режим доступа: https://www.section.io/engineering-education/why-python-is-good-for-machine-learning/ – 10.03.2022

4 В чем особенности MongoDB и когда эта база данных вам подходит: руководство для новичков [Электронный ресурс]. – Режим доступа: https://mcs.mail.ru/blog/osobennosti-mongodb-kogda-baza-dannyh-vam-podhodit – 10.03.2022

5 Онлайн-руководство по MongoDB [Электронный ресурс]. – Режим доступа: https://metanit.com/nosql/mongodb/1.1.php – 15.05.2022

6 Что такое машинное обучение и как оно работает [Электронный ресурс]. – Режим доступа: https://trends.rbc.ru/trends/industry/60c85c599a7947f5776ad409 – 15.05.2022

7 Валидность верстки [Электронный ресурс]. – Режим доступа: https://serpstat.com/ru/blog/kak-proverit-validnost-verstki – 15.05.2022

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Адаптивная верстка [Электронный ресурс]. – Режим доступа: https://html5book.ru/adaptivnaya-vyorstka-sayta/ – 15.04.2022

2 Айсберг Big Data: почему мы видим лишь вершину. [Электронный ресурс]. – Режим доступа: http://sap-technology.rbc.ru/big-data.html. – 01.02.2022.

3 Безопасность веб-приложений. – СПб.: Питер, 2022. – 336 с.: ил.

4 Блок-схема [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA-%D1%81%D1%85%D0%B5%D0%BC%D0%B0 – 10.03.2022

5 Большие данные (Big Data) [Электронный ресурс]. – Режим доступа: https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%91%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_(Big_Data) – 02.02.2022

6 В чем особенности MongoDB и когда эта база данных вам подходит: руководство для новичков [Электронный ресурс]. – Режим доступа: https://mcs.mail.ru/blog/osobennosti-mongodb-kogda-baza-dannyh-vam-podhodit – 10.03.2022

7 Валидность верстки [Электронный ресурс]. – Режим доступа: https://serpstat.com/ru/blog/kak-proverit-validnost-verstki – 15.05.2022

8 Введение в pandas: анализ данных на Python [Электронный доступ] – Режим доступа: https://khashtamov.com/ru/pandas-introduction/ – 15.05.2022

9 Веб-скрейпинг [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%BA%D1%80%D0%B5%D0%B9%D0%BF%D0%B8%D0%BD%D0%B3 – 10.03.2022

10 Глубокое обучение [Электронный ресурс]. – Режим доступа: https://www.baslerweb.com/ru/vision-campus/otrasli-i-zadachi/chto-takoe-glubokoe-obuchenie/ – 15.05.2022

11 Глубокое обучение [Электронный ресурс]. – Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=Глубокое_обучение – 15.05.2022

12 Грас Дж. Data Science. Наука о данных с нуля: Пер. с англ. – СПб.: БХВ-Петербург, 2019. – 336с.: ил.

13 Документация по Beautiful Soup [Электронный ресурс]. – Режим доступа: https://www.crummy.com/software/BeautifulSoup/bs4/doc.ru/bs4ru.html – 15.04.2022

14 Жизненный цикл модели машинного обучения [Электронный ресурс]. – Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=Жизненный_цикл_модели_машинного_обучения – 10.03.2022

15 Заголовки HTTP [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/HTTP/Headers – 10.03.2022

16 Киберспорт [Электронный ресурс]. – Режим доступа: https://znaki.fm/esports/ - 01.02.2022.

17 Мега-Учебник Flask [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/193242/ – 15.05.2022

18 Модели машинного обучения [Электронный ресурс]. – Режим доступа: https://vc.ru/ml/126354-modeli-mashinnogo-obucheniya-obyasnyaempyatiletnemu-rebenku – 15.05.2022

19 Облегчаем себе жизнь с помощью BeautifulSoup4 [Электронный доступ] – Режим доступа: https://habr.com/ru/post/544828/ – 15.05.2022

20 Онлайн-руководство по MongoDB [Электронный ресурс]. – Режим доступа: https://metanit.com/nosql/mongodb/1.1.php – 15.05.2022

21 Парсинг. Что это и где используется [Электронный ресурс]. – Режим доступа: https://ipipe.ru/info/parsing – 10.03.2022

22 Разработка на JavaScript. – СПб.: Питер, 2021. – 320 с.: ил.

23 Свечников С.Н. Разработка информационной системы предсказания рейтинга команд в киберспорте // материалы XXII региональной научно-практической конференции (20 мая 2021 года) – Благовещенск: Изд-во БГПУ, 2021. – 964 с. – С. 784-785.

24 Свечников С.Н. Разработка информационной системы предсказания рейтинга команд в киберспорте // материалы XXIII региональной научно-практической конференции (Благовещенск, 24 мая 2022 г.). [В 4 т.]. Т. 4. Технические науки. Физико-математические науки. Информационные технологии. Химические науки. – Благовещенск: Дальневосточный ГАУ, 2022. – 355 с. – С. 211-212.

25 Современный скрапинг веб-сайтов с помощью Python. 2-е межд. изд. – СПб.: Питер, 2021. – 336 с.: ил.

26 Создаем нейронную сеть.: Пер. с англ. – СПб.: ООО «Диалектика», 2019. – 272 с.: ил.

27 Стандарт организации. Оформление выпускных квалификационных и курсовых работ (проектов) [Электронный ресурс] / АмГУ ; разраб. Л. А. Проказина, Н. А. Чалкина, С. Г. Самохвалова. - Введ. с 05.04.2018. - Благовещенск : [б. и.], 2018. - 75 с. – 03.02.2022

28 Что такое big data: зачем нужны большие данные, как их собирают и обрабатывают [Электронный ресурс]. – Режим доступа: https://mcs.mail.ru/blog/big-data-vse-govoryat-no-malo-kto-shchupal - 03.02.2022

29 Что такое Scikit Learn - гайд по популярной библиотеке Python для начинающих [Электронный ресурс] – Режим доступа: https://datastart.ru/blog/read/chto-takoe-scikit-learn-gayd-po-populyarnoy-biblioteke-python-dlya-nachinayuschih – 15.05.2022

30 Что такое большие данные? [Электронный ресурс]. – Режим доступа: https://www.oracle.com/ru/big-data/what-is-big-data/#use-cases – 02.02.2022

31 Что такое машинное обучение и как оно работает [Электронный ресурс]. – Режим доступа: https://trends.rbc.ru/trends/industry/60c85c599a7947f5776ad409 – 15.04.2022

32 Ын Анналин, Су Кеннет Теоретический минимум по Big Data. Всё, что нужно знать о больших данных. – СПб.: Питер, 2019. – 208 с.: ил. – 02.02.2022

33 BeautifulSoup – парсинг HTML в Python на примерах [Электронный доступ] – Режим доступа: https://python-scripts.com/beautifulsoup-html-parsing – 15.04.2022

34 Big Data [Электронный ресурс]. – Режим доступа: https://www.calltouch.ru/glossary/big-data/ - 02.02.2022

35 Big Data данные [Электронный ресурс]. – Режим доступа: https://www.it.ua/ru/knowledge-base/technology-innovation/big-data-bolshie-dannye – 03.02.2022

36 Big Data от А до Я. Часть 1: Принципы работы с большими данными, парадигма MapReduce [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/267361/ - 02.02.2022

37 CSS – селекторы [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/Web/CSS/CSS_Selectors – 15.04.2022

38 Flask [Электронный ресурс]. – Режим доступа: https://flask.palletsprojects.com/en/2.1.x/foreword/ – 15.05.2022

39 Jquery [Электронный ресурс] – Режим доступа: https://jquery.com/ – 17.05.2022

40 Machine Learning (машинное обучение): что это такое [Электронный ресурс]. – Режим доступа: https://www.bigdataschool.ru/wiki/machine-learning – 18.05.2022

41 MongoDB [Электронный ресурс] – Режим доступа: https://www.mongodb.com/ – 19.04.2022

42 MongoDB: полное руководство. Мощная и масштабируемая система управления базами данных. 3-я редакция / пер. с англ. Д. А. Беликова – М.: ДМК Пресс, 2020. – 540 с.: ил.

43 NumPy [Электронный доступ] – Режим доступа: https://numpy.org/ – 15.05.2022

44 NumPy в Python. Часть 1 [Электронный доступ] – Режим доступа: https://habr.com/ru/post/352678/ – 15.03.2022

45 NumPy, часть 1: начало работы [Электронный доступ] – Режим доступа: https://pythonworld.ru/numpy/1.html – 15.03.2022

46 Pandas [Электронный ресурс] – Режим доступа: https://pandas.pydata.org/ – 15.03.2022

47 Python библиотеки для Data Science [Электронный ресурс]. – Режим доступа: https://tproger.ru/translations/top-10-python-bibliotek-dlja-datascience/ – 15.05.2022

48 Python. Библиотека Pandas [Электронный доступ] – Режим доступа: https://istories.media/workshops/2021/03/05/python-biblioteka-pandas-chast-1/ – 15.05.2022

49 Requests: HTTP for humans [Электронный ресурс]. – Режим доступа: https://docs.python-requests.org/en/latest/ – 15.05.2022

50 TensorFlow для глубокого обучения: Пер. с англ. / Б. Рамсундар, Р. Б. Заде. – СПб.: БХВ-Петербург, 2019 – 256 с.: ил – 12.03.2022

51 When to Use NoSQL and MongoDB [Электронный ресурс]. – Режим доступа: https://www.fivetran.com/blog/when-to-use-nosql-mongodb – 10.03.2022

52 Why Python is Good for Machine Learning [Электронный ресурс]. – Режим доступа: https://www.section.io/engineering-education/why-python-is-good-for-machine-learning/ – 10.03.2022

ПРИЛОЖЕНИЕ А