

Минобрнауки России

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Санкт-Петербургский государственный технологический институт

(технический университет)»

|  |  |  |
| --- | --- | --- |
| Направление | 09.03.02 | Информационные системы и технологии |
| Факультет |  | Информационных технологий и управления |
| Кафедра |  | Системного анализа и информационных технологий |
| ***Учебная дисциплина*** |  | ***Методы искусственного интеллекта*** |
| Курс 3 |  | Группа 428 |

**КУРСОВАЯ РАБОТА**

**Тема:**

**Технологии искусственного интеллекта в предсказании матчей по доте 2.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | |  | | --- | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |   (дата, подпись) | |  | | --- | | Коготков М.А. |   (фамилия, инициалы) |
| Руководитель, доцент | |  | | --- | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |   (дата, подпись) | |  | | --- | | Мусаев А.А. |   (фамилия, инициалы) |
| Оценка за курсовую работу | |  | | --- | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |   (оценка) | |  | | --- | |  |   (подпись руководителя) |

Санкт-Петербург

2024



Минобрнауки России

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Направление | **09.03.02** | | Информационные системы и технологии | |
| Направленность | |  | | Информационные системы и технологии | |
| Факультет | |  | | Информационных технологий и управления | |
| Кафедра | |  | | Системного анализа и информационных технологий | |
| ***Учебная дисциплина*** | |  | | ***Методы искусственного интеллекта*** | |
| Курс 3 | |  | | Группа 428 | |
| Студент | |  | | Коготков Михаил Алексеевич | |

**Тема:** Использование технологии искусственного интеллекта для предсказания результатов матчей по игре Dota 2.

***Перечень вопросов, подлежащих разработке***

Данное исследование направлено на изучение применения искусственного интеллекта в киберспорте, включая его основные принципы, области применения (предсказание матчей, рассмотрение различных команд) и перспективы развития.

***Требования к оформлению***

Курсовая работа должна быть оформлена в соответствии с требованиями СТО СПбГТИ 044 –2012 КС УКДВ. Виды учебных занятий. Курсовой проект. Курсовая работа. Общие требования.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Дата выдачи задания** | |  | **2024 г.** | | |
| **Срок представления к защите** | |  | **2024 г.** | | |
| **Зав. кафедрой** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |  | д.т.н., профессор | |
| Мусаев А.А. | |
|  | (подпись) | | |  |  | |
| **Руководитель,**  **Должность** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |  | к.т.н., доцент  Мусаев А.А. | |
|  | (подпись) | | |  |  | |
| **Задание принял**  **к выполнению** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |  | Коготков М.А. | |
|  | (подпись) | | |  |  | |

**Содержание**

Введение…………………………………………………………………………4

1. Теоретический обзор…………………………………………………………6

* 1. Понятие искусственного интеллекта и его виды………………………….6
  2. Машинное обучение основные принципы и алгоритмы…………………6
  3. Основные механик игры Dota 2……………………………………………8
  4. Роли игроков………………………………………………………………..12

2. Сбор данных………………………………………………………………….14

2.1. Источники данных…………………………………………………………14

2.2. Структура данных………………………………………………………….14

3. Разработка искусственного интеллекта…………………………………….15

3.1 Выбор модели искусственного интеллекта……………………………….15

3.2 Реализации искусственного интеллекта…………………………………..16

3.3 Разработка пользовательского интерфейса……………………………….19

Заключение……………………………………………………………………..26

Список литературы…………………………………………………………….27

Приложение…………………………………………………………………….28

**Введение**

Киберспорт стремительно развивается, завоевывая признание как среди зрителей, так и среди профессиональных игроков. С каждым годом все больше людей интересуются киберспортивными соревнованиями, что приводит к увеличению числа турниров и призовых фондов. В этом контексте предсказания результатов матчей становятся важным инструментом как для команд, так и для аналитиков, позволяя оптимизировать стратегии и принимать обоснованные решения.

Киберспорт — это форма соревнования, в которой игроки соревнуются друг с другом в видеоиграх. С момента своего появления он прошел путь от нишевого увлечения до полноценной индустрии, генерирующей миллиарды долларов в год. Особое внимание уделяется организационным аспектам: формированию команд, проведению крупных турниров и созданию профессиональных лиг. Киберспорт уже стал частью массовой культуры, привлекая миллионы зрителей по всему миру и обеспечивая значительное внимание со стороны спонсоров и медиа.

Dota 2, разработанная компанией Valve, является одной из самых популярных игр в жанре MOBA. С момента своего релиза в 2013 году она завоевала огромную аудиторию благодаря захватывающему игровому процессу, высокому уровню стратегии и постоянным обновлениям. Профессиональные турниры, такие как The International, привлекают миллионы зрителей и предлагают внушительные призовые фонды, что подтверждает статус Dota 2 как одной из ведущих игр в мире киберспорта. Целью игры разрушить вражеский трон, для этого необходимо выбирать правильных героев, покупать правильные предметы и придерживаться правильной тактики игры.

Предсказания результатов матчей играют важную роль в киберспорте и ставках. Они помогают командам разрабатывать стратегии, основанные на анализе предыдущих матчей и текущих трендов, а также дают возможность беттерам делать обоснованные ставки. Использование искусственного интеллекта и машинного обучения для предсказания вероятности победы на основе статистики и анализа выбора героев открывает новые горизонты в стратегии и аналитике. В условиях высокой конкуренции такие инструменты становятся необходимыми для достижения успеха как на профессиональной арене, так и в мире ставок.

**Цели и задачи работы**

* **Основная цель**: Разработка модели искусственного интеллекта для предсказания результатов матчей.
* **Задачи**:
  + Сбор и обработка статистических данных о матчах.
  + Создание и обучение модели машинного обучения.

**1.Теоретический обзор**

**1.2 Понятие искусственного интеллекта и его виды.**

Искусственный интеллект — это область компьютерных наук, занимающаяся созданием программ и систем, способных выполнять задачи, которые требуют человеческого интеллекта [1]. Эти задачи могут включать в себя:

* Обучение (например, адаптация к новым данным)
* Решение проблем (например, нахождение оптимального решения)
* Распознавание образов (например, идентификация лиц на фотографиях)
* Обработка естественного языка (например, понимание и генерация текста)

**Виды искусственного интеллекта:**

Узкий ИИ (или слабый ИИ):

Это системы, созданные для выполнения конкретных задач. Узкий ИИ не обладает сознанием или пониманием, он просто выполняет заданные алгоритмы.

Примеры:

* Голосовые помощники (Siri, Alexa)
* Рекомендательные системы (Netflix, Amazon)
* Игровые ИИ (игры, такие как шахматы и тд)

Общий ИИ (или сильный ИИ):

Это гипотетическая форма ИИ, которая могла бы выполнять любые интеллектуальные задачи на уровне человека. На данный момент такой ИИ не существует, и его создание является одной из главных целей в области ИИ [1].

Общий ИИ должен был бы обладать способностью к самосознанию, пониманию и адаптации к новым ситуациям.

**1.3 Машинное обучение (МО) и его типы**

Машинное обучение (ML) — это область искусственного интеллекта, которая позволяет системам автоматически учиться и улучшаться на основе опыта без явного программирования. В последние годы машинное обучение стало ключевым инструментом в различных областях, таких как медицина, финансы, маркетинг и многое другое. В этой статье мы рассмотрим основные виды и типы машинного обучения, чтобы помочь вам лучше понять, как они работают и где могут быть применены [2].

**Типы машинного обучения:**

Обучение с учителем:

Обучение с учителем (Supervised Learning) — это тип машинного обучения, при котором модель обучается на размеченных данных. Это означает, что каждый обучающий пример в наборе данных имеет соответствующую метку или целевое значение. Цель модели — научиться предсказывать метки для новых, невиданных данных [2].

Примеры задач обучения с учителем:

* Определение категории, к которой принадлежит объект. Например, распознавание рукописных цифр (0-9) или классификация писем как "спам" или "не спам". Классификация также может применяться в медицине для диагностики заболеваний на основе симптомов пациента.
* Предсказание непрерывного значения. Например, прогнозирование цен на недвижимость или предсказание температуры на следующий день. Регрессия также используется в экономике для прогнозирования роста ВВП или в маркетинге для оценки эффективности рекламных кампаний.

Примеры алгоритмов обучения с учителем:

* Линейная регрессия
* Логистическая регрессия
* Деревья решений
* Метод опорных векторов (SVM)
* Нейронные сети

Обучение без учителя

Обучение без учителя (Unsupervised Learning) — это тип машинного обучения, при котором модель обучается на неразмеченных данных. В этом случае модель пытается найти скрытые структуры или закономерности в данных без явных меток [2].

Примеры задач обучения без учителя:

* Группировка объектов в кластеры на основе их сходства. Например, сегментация клиентов на основе их покупательского поведения. Кластеризация также может использоваться в биологии для группировки генов с похожими функциями или в маркетинге для создания целевых аудиторий.
* Уменьшение количества переменных в данных для упрощения анализа. Например, метод главных компонент (PCA) используется для визуализации многомерных данных. Снижение размерности также полезно для уменьшения шума в данных и улучшения производительности алгоритмов машинного обучения.

Примеры алгоритмов обучения без учителя

* K-средних (K-means)
* Иерархическая кластеризация
* Метод главных компонент (PCA)
* Ассоциативные правила

**1.4 Основные игровые механики**

До начала матча игроки выбирают себе героя из доступных. У каждого героя есть несколько аспектов, которые раскрывают разные стили игры за персонажа. Перед началом матча игрок должен выбрать один аспект, исходя из конкретной ситуации: героев соперника, позиции в команде и предпочтительного характера геймплея [6]. Выбор аспекта в Dota 2 доступен всем игрокам на стадии планирования — после экрана загрузки на карту изменить его будет нельзя. При этом соперники не увидят, какой именно аспект вы решили использовать, пока не начнется матч.

(Рисунок 1)

Изображение выглядит как текст, снимок экрана, Шрифт, графический дизайн

Автоматически созданное описание

Рисунок 1 – аспекты героя

Также до начала игры можно выбрать косметические предметы для персонажа. Эти предметы никак не влияют на игровой процесс, лишь дают визуальные эффекты.

Атрибуты — это главные характеристики всех героев, которые определяют большинство изменяющихся параметров. Таких атрибутов (характеристик) три: сила, ловкость и интеллект. Они увеличиваются с повышением уровня героя, выбором таланта, покупкой некоторых предметов или использованием некоторых способностей [7]. Иллюзии также зависят от атрибутов. Также атрибуты в Dota 2 влияют на максимальное количество здоровья, маны, броню и магическое сопротивление. (Рисунок 2)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 2 – атрибуты героя

Способности — это уникальные навыки, которыми могут обладать герои и крипы. В зависимости от принципа действия, все способности делятся на активные и пассивные. Существует большое количество различных способностей, от простых пассивных до разрушительных энергетических взрывов, способных сотрясать землю. Герои могут обладать четырьмя и более способностями, одна из которых является особой. Каждый раз, когда герой повышает свой уровень, он может улучшить одну из своих способностей или выучить новую.

Активные способности при использовании потребляют часть маны героя. После использования активной способности включается таймер её перезарядки. Таким образом, должно пройти некоторое время, прежде чем данной способностью можно будет воспользоваться снова [8]. (Рисунок 3)

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 3 – способности героя

Предметы — внутриигровое снаряжение героев, которое может давать героям дополнительные характеристики и специальные способности. Их можно купить в нескольких лавках на игровой карте. Некоторые предметы низкого разряда складываются в предметы высшего разряда, с помощью рецепта. Герои имеют шесть ячеек для предметов в их инвентаре, три ячейки в ранце, отдельную ячейку для нейтральных предметов и отдельную ячейку для Town Portal Scroll, а также дополнительных шесть в их тайнике. Курьер может поднимать и отсылать предметы их владельцу [9]. (Рисунок 4)



Рисунок 4 – предметы игры

Опыт и золото являются самыми важными для победы в игре. Золото — валюта, используемая для покупки предметов или мгновенного возрождения героя. Золото можно заработать путём убийства героев, крипов или при уничтожении построек. Золото игрока делится на надёжное золото и ненадёжное золото. Пассивный прирост золота каждому игроку, так же как и золото за активацию рун богатства, считается надёжным золотом, в то время как золото со всех других источников считается ненадёжным золотом. Благодаря опыту ваш герой может поднять уровень, увеличивая свои характеристики и получая доступ к новым способностям или талантам. В начале игры все персонажи стартуют с первым уровнем и одним очком способности. Зарабатывая опыт благодаря убийству крипов и героев, контролю аванпостов.

**1.5 Роли игроков**

Всего в игре одновременно участвуют десять человек, соответственно они играют за две команды силы света и силы тьмы. В начале игры игроки распределяются на три линии, по двое на легкую и сложную линии соответственно, и один на центральный коридор. После примерно 10 минут игроки уходят с линий на другие части карты, однако их роли остаются. Вот основные роли игроков в игре:

* Керри — это персонажи, главная задача которых – заработок золота и нанесение урона как героям, так и зданиям противника. В преимущественном большинстве случаев керри слабы на начальной стадии, из-за чего им требуется помощь саппортов на линии. Ключевая их задача заключается в том, чтобы зарабатывать золото, убивая как можно больше крипов на протяжении всей игры, в связи с чем главный навык для игроков на этой позиции – ластхит, то есть умение хорошо добивать крипов на линии. При этом они чаще всего имеют простые способности, а основной урон наносят своей атакой.
* Мидер – это герой центральной линии, который по своему принципу игры крайне схож с керри, но имеет несколько отличительных особенностей. Чаще всего он вносит больше пользы за счет использования своих способностей, в связи с чем игрокам на этой роли важно уметь не только добивать крипов, но и правильно использовать скиллы на всех персонажах.
* Оффлейнеры стоят на сложной линии и могут преследовать две задачи – максимально препятствовать фарму керри противника или помогать своей команде. На эту позицию чаще всего берутся персонажи, которые могут самостоятельно сопротивляться или даже создавать дискомфорт сразу для двух или трех противников. Помимо этого, на данную роль берутся герои, которые могут компенсировать какие-то недостатки пары керри+мидер – контроль или урон.
* Роумер – это герой поддержки, который отличается от обычного саппорта тем, что он чаще всего покупает меньше расходников, так как постоянно помогает на всех линиях, где это требуется, а также должен вносить большую пользу во время драки. Помимо этого, если появляется возможность заработать золото на определенной линии, чаще всего этим занимается именно игрок четвертой позиции, чтобы во время драки вносить еще больше пользы. Игрок на этой позиции должен уметь приносить максимум пользы за счет использования своих способностей.
* Главная задача саппорта – это помощь своему керри в фарме, а также расстановка вардов по карте. Герой должен иметь простые в использовании способности, чтобы была возможность выбросить их перед смертью и принести при этом пользу команде [4].

**2. Сбор данных**

**2.1. Источники данных**

Официальные API Dota 2:

Valve предоставляет API, который позволяет разработчикам получать доступ к различной статистике игры, включая информацию о матчах, героях, игроках и других аспектах Dota 2. С помощью API можно извлекать данные о текущих и завершенных матчах, а также получать информацию о героях и их способностях.

Платформы для анализа матчей:

OpenDota: это открытая платформа, которая предоставляет доступ к данным о матчах Dota 2. Она предлагает API и интерфейс для анализа и визуализации статистики, включая подробные отчеты о матчах, производительности героев и игроков.

Dotabuff: это популярный сайт, который предоставляет подробную статистику о матчах, игроках и героях. Dotabuff предлагает различные аналитические инструменты, такие как сравнение игроков, статистика по героям и многое другое.

Сайты с готовыми датасетами kugglehub и другие.

**2.2. Структура данных**

В данном случае данные представлены в формате parquet. Данные содержат порядка 20000 данных о матчах.

Parquet — это бинарный, колоночно-ориентированный формат хранения больших данных, изначально созданный для экосистемы Hadoop, позволяющий использовать преимущества сжатого и эффективного колоночно-ориентированного представления информации. Паркет позволяет задавать схемы сжатия на уровне столбцов и добавлять новые кодировки по мере их появления [5].

Данные представляют собой таблицу с записями о матчах, как числовыми, так и текстовыми. Представлены следующие данные, названия команд, имена игроков, герои, которые участвовали в игре, количество опыта и золота в минуту, общее количество убийств, смертей, продолжительность игры в секундах, время первой с момента начала матча в секундах и версия игры.

**3. Разработка искусственного интеллекта**

**3.1 Выбор модели искусственного интеллекта**

Случайный лес - это ансамблевый метод машинного обучения, основанный на комбинировании множества решающих деревьев. Он представляет собой набор деревьев, где каждое дерево строится независимо от других деревьев с использованием случайной подвыборки данных и случайного подмножества признаков.

Процесс построения случайного леса начинается с выбора случайной подвыборки обучающих данных из исходного набора данных. Затем для каждого дерева в лесу происходит построение дерева на основе этой случайной подвыборки. В процессе построения каждого дерева используется случайное подмножество признаков. Количество признаков, выбираемых для каждого дерева, определяется параметром, называемым гиперпараметром "число признаков".

Когда классификация или регрессия выполняются с помощью случайного леса, каждое дерево в лесу вносит свой вклад в окончательное решение. В случае классификации оно может голосовать за наиболее популярный класс среди всех деревьев, а в случае регрессии - среднее значение или медиану предсказаний, полученных от всех деревьев.

Случайный лес является мощным алгоритмом для решения задач классификации и регрессии. Он имеет ряд преимуществ, включая способность обрабатывать как числовые, так и категориальные признаки, робастность к выбросам и шуму в данных, устойчивость к переобучению и способность обрабатывать большие объемы данных. Он также позволяет оценить важность признаков и использовать их для дальнейшего анализа.

Однако случайный лес также имеет свои ограничения. Он может быть достаточно вычислительно сложным и требовательным к памяти, особенно при больших размерах данных и большом количестве деревьев. Кроме того, интерпретация случайного леса может быть сложной, поскольку результат представляет собой комбинацию множества деревьев.

**3.2 Реализации искусственного интеллекта**

Для обучения модели использовались id игроков и героев, время матча, опыт и золото игроков в минуту, КДА каждого игрока. Для обучения мы получаем первые n значений из таблицы данных, затем заполняем пропуски. Далее выбираем целевой столбец “winner\_id” далее в коде происходит замена id команд на 1 или 2 в случае победы radiant или dire соответственно. Также данные разделяются на обучающие и тестовые для оценки общей точности модели (80% данных для обучения и 20% тестовых данных, параметр random\_state дает одинаковое распределение данных. После чего происходит обучение модели RandomForestClassifier. (Рисунок 5)



Рисунок 1 – алгоритм обучения модели

Начальная таблица данных имеет не нужные для обучения модели столбцы такие, как “dire\_hero\_1\_name”, “league\_id” и т.д., так как многие из этих данных дублируются в столбцы id героя или игрока. Соответственно перед обучением модели все эти данные убираются из таблицы. (Рисунок 6) Загрузка таблицы данных из формата .parquet происходит с помощью библиотеки pyarrow, и метода read\_table. После загрузки данных в датафрейм происходит создание таблиц для игроков, команд и героев. Эти таблицы заполняются с помощью метода concat из библиотеки pandas, с помощью метода из той же библиотеки drop\_duplicates() удаляются повторяющиеся значения. (Рисунок 7)

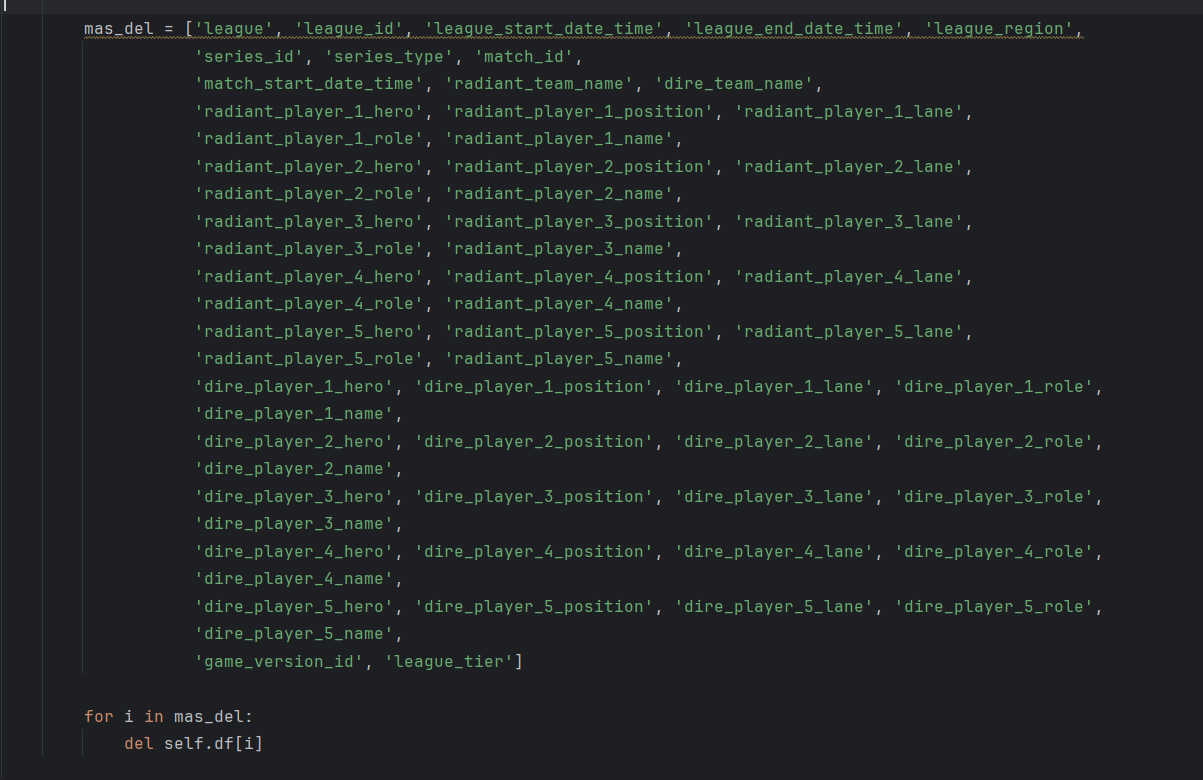


Рисунок 6 – удаление ненужных данных



Рисунок 7 – загрузка, удаление дубликатов и создание таблиц для команд, игроков и героев

Можно сохранить или загрузить уже обученную модель с помощью библиотеки joblib.(Рисунок 8)

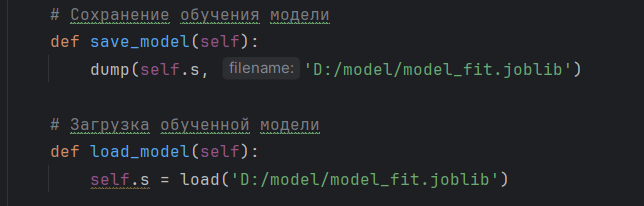


Рисунок 8 – сохранение и загрузка обученной модели

После обучения модели можно проверить результат реального матча, но так в до окончание матча не известны ни КДА игроков, ни их общее количество денег. А известны герои и иногда имена игроков и названия команд, необходимо заполнить отсутствующие данные пропусками. (Рисунок 9) В случае отсутствия данных об игроке или команде в соответствующей таблице это значение ставиться None. Если же данные есть, то заменяться на значение id игрока или команды.



Рисунок 9 – заполнение пропусков во входных данных

После подготовки данных для предсказания результата матча уже обученной

моделью. Модель возвращает 1 в случае победы radiant и 2 после победы dire. Также программа может вернуть общую точность модели. (Рисунок 10)

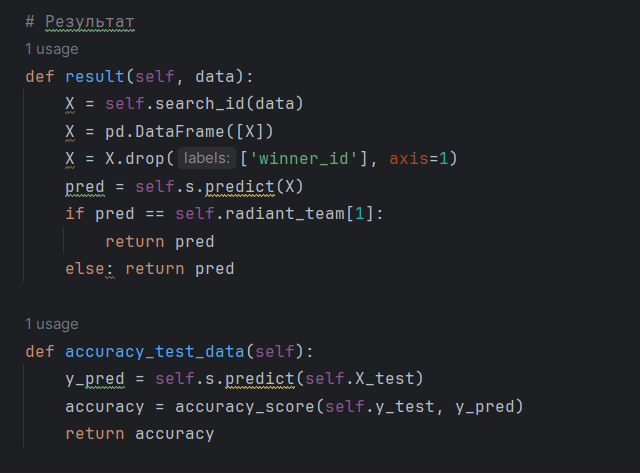


Рисунок 10 – возврат результата

**3.3 Разработка пользовательского интерфейса**

Для реализации пользовательского интерфейса использовался телеграмм бот, со следующими функциями. (Рисунок 11) Пример работы бота на Рисунках 12-14. Для реализации бота использовались библиотеки telebot и spacy.

Изображение выглядит как текст, электроника, снимок экрана, число

Автоматически созданное описание

Рисунок 11 – функции бота

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 13 – пример выполнения функции вывод точности

Изображение выглядит как текст, Человеческое лицо, одежда, человек

Автоматически созданное описание

Рисунок 14 – пример предсказания результатов матча

Для реализации загружаем модель SpaCy для работы с русским языком. Вызываем экземпляр класса dota\_ai (реализующего модель для предсказания матчей игры Dota 2). Затем происходит обучение модели на данных с размером выборки n (ai.train(n)). Далее происходит получение списка доступных героев из ai.df\_hero. Создается словарь для хранения пользовательских данных. Затем происходит инициализация Telegram-бота с заданным токеном. (Рисунок 15)

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 15 – реализация пользовательского интерфейса

Далее рассмотрим функцию обработки команды /start. При запуске бота создаётся клавиатура с кнопками: "Начать использование" — для перехода к функционалу бота. И "Помощь" — для отображения информации о функциях бота. (Рисунок 16)

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 16 - реализация функции обработки команды /start

Данная функция обрабатывает текстовые сообщения от пользователя для этого, текст приводится к нижнему регистру (lower) и обрабатывается моделью SpaCy. В обработке SpaCy разбивает предложение на токены и находит базовые формы слов (леммы). Затем если текст содержит слово "помощь", вызывается функция show\_help. (Рисунок 18) Эта функция отправляет пользователю информацию по использованию бота. Если текст содержит слово "начать", вызывается функция start\_usage. (Рисунок 19) Данная функция создает клавиатуру с действиями: "Вывод точности модели", "Предсказание результата матча" и " Список героев". Если текст содержит слово "точность", отправляется сообщение с точностью модели, вызвав метод accuracy\_test\_data(). В этом методе пользователь вводит данные для предсказания: происходит преобразование данных через функцию text\_parser.parse\_input\_text\_to\_data. Если эти данные успешно обрабатываются, то предсказывается победитель (ai.result). В противном случае пользователю отправляется сообщение об ошибке. Также при нахождении в тексте слова "Герой" выводит пользователю список доступных героев.Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 17 – функция обработки сообщений

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 18 – функция show\_help

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 19 – функция start\_usage

В функции parse\_input\_text\_to\_data происходит обработка текста через SpaCy и инициализации словаря (Рисунок 20)

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 20 - обработка текста и инициализация словаря

Далее в функции происходит заполнение словаря. (Рисунок 20) Входной текст предварительно разбивается на предложения с использованием spaCy. Каждый элемент в цикле for sent in doc.sents представляет отдельное предложение из текста. Если в текущем предложении встречается фраза "Команда Radiant", это означает, что оно содержит название команды Radiant. Название извлекается с помощью разбиения строки по символу ":". Берется часть строки после двоеточия, убираются лишние пробелы и точка в конце строки. Аналогично, для фразы "Команда Dire". После извлечения названия текущая команда устанавливается в Radiant или Dire для дальнейшей обработки. Если в предложении есть фраза "Игроки:", это указывает на перечисление игроков и героев. Список игроков и героев разбивается на элементы с помощью split(", "), чтобы каждая пара обрабатывалась отдельно. Для каждой пары проверяется наличие слова "на" (что подтверждает формат "Игрок на Герой"). Если пара корректна, она делится на имя игрока и имя героя с помощью split(" на ").

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 20 – заполнение словаря

**Заключение**

В ходе выполнения этой курсовой работы была разработана модель искусственного интеллекта, способная предсказывать результаты матчей в популярной игре Dota 2. Были изучены основные принципы работы искусственного интеллекта и машинного обучения, а также проанализировали игровые механики и ключевые параметры, которые влияют на исход матчей.

Для обучения модели использовалась подготовленная база данных о результатах уже прошедших матчей, включающая в себя данные из официальных API и аналитических платформ.

В качестве алгоритма был выбран случайный лес, так как он хорошо справляется с учётом сложных взаимосвязей между игровыми параметрами. В результате была получена модель с точностью предсказаний в 95%, что можно объяснить особенностями игры, где итоговый результат матча после его завершения не сложно определить на основе объективных параметров.

Созданный инструмент может быть полезен для игроков, тренеров и аналитиков. Он позволяет лучше понимать текущие игровые тренды и принимать более взвешенные решения. В будущем можно улучшить модель, добавив больше данных и оптимизировав алгоритмы.

Эта работа подтвердила, что искусственный интеллект имеет огромный потенциал в киберспорте. Современные технологии анализа данных и машинного обучения позволяют не только улучшать качество аналитики, но и делать игру интереснее, создавая новые способы взаимодействия игроков с аналитическими инструментами.

**Список литературы.**

1. Искусственный интеллект: // Sky.pro. 2023. Электронный ресурс. URL: https://sky.pro/wiki/python/chto-takoe-iskusstvennyj-intellekt/ (дата обращения: 25.10.2023).
2. Виды и типы машинного обучения: // Sky.pro. 2023. Электронный ресурс. URL: https://sky.pro/wiki/python/vidy-i-tipy-mashinnogo-obucheniya/ (дата обращения: 25.10.2023).
3. Механика: // Dota2.fandom.com. 2023. Электронный ресурс. URL: https://dota2.fandom.com/ru/wiki/%D0%9C%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D0%BA%D0%B0 (дата обращения: 25.10.2023).
4. Роли в Dota 2: правильный выбор позиции: // Anylvl.com. 2023. Электронный ресурс. URL: https://anylvl.com/ru/blog/statja-roli-v-dota2-pravilniy-vibor-positcii (дата обращения: 25.10.2023).
5. Parquet: // Bigdataschool.ru. 2023. Электронный ресурс. URL: https://bigdataschool.ru/wiki/parquet (дата обращения: 25.10.2023).
6. Аспекты в Dota 2: что это такое и как их выбирать?: // Dota2.ru. 2023. Электронный ресурс. URL: https://dota2.ru/faq/219-aspekty-v-dota-2-cto-eto-takoe-i-kak-ih-vybirat/ (дата обращения: 25.10.2023).
7. Атрибуты: // Dota2.fandom.com. 2023. Электронный ресурс. URL: https://dota2.fandom.com/ru/wiki/%D0%90%D1%82%D1%80%D0%B8%D0%B1%D1%83%D1%82%D1%8B (дата обращения: 25.10.2023).
8. Способности: // Dota2.fandom.com. 2023. Электронный ресурс. URL: https://dota2.fandom.com/ru/wiki/%D0%A1%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%BD%D0%BE%D1%81%D1%82%D0%B8 (дата обращения: 25.10.2023).
9. Предметы: // Dota2.fandom.com. 2023. Электронный ресурс. URL: https://dota2.fandom.com/ru/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D1%8B (дата обращения: 25.10.2023).

**Приложение.**

Dota\_ai.py

from joblib import dump, load

import pandas as pd

import numpy as np

import pyarrow.parquet as pq

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

class dota\_ai():

def \_\_init\_\_(self):

table = pq.read\_table('dota2\_matches.parquet')

self.df = table.to\_pandas()

self.df\_player\_name = pd.DataFrame()

self.df\_player\_id = pd.DataFrame()

self.df\_hero = pd.DataFrame()

self.df\_hero\_id = pd.DataFrame()

self.df\_team = pd.DataFrame()

self.df\_team\_id = pd.DataFrame()

self.df\_team = pd.concat([self.df\_team, self.df['radiant\_team\_name']], ignore\_index=True).drop\_duplicates()

self.df\_team = pd.concat([self.df\_team, self.df['dire\_team\_name']], ignore\_index=True).drop\_duplicates()

self.df\_team\_id = pd.concat([self.df\_team\_id, self.df['radiant\_team\_id']], ignore\_index=True).drop\_duplicates()

self.df\_team\_id = pd.concat([self.df\_team\_id, self.df['dire\_team\_id']],ignore\_index=True).drop\_duplicates()

for i in range(1,6):

self.df\_player\_name = pd.concat([self.df\_player\_name, self.df[f'radiant\_player\_{i}\_name']],ignore\_index=True).drop\_duplicates()

self.df\_player\_name = pd.concat([self.df\_player\_name, self.df[f'dire\_player\_{i}\_name']], ignore\_index=True).drop\_duplicates()

self.df\_player\_id = pd.concat([self.df\_player\_id, self.df[f'radiant\_player\_{i}\_id']],ignore\_index=True).drop\_duplicates()

self.df\_player\_id = pd.concat([self.df\_player\_id, self.df[f'dire\_player\_{i}\_id']], ignore\_index=True).drop\_duplicates()

self.df\_hero = pd.concat([self.df\_hero, self.df[f'radiant\_player\_{1}\_hero']], ignore\_index=True).drop\_duplicates()

self.df\_hero = pd.concat([self.df\_hero, self.df[f'dire\_player\_{1}\_hero']], ignore\_index=True).drop\_duplicates()

self.df\_hero\_id = pd.concat([self.df\_hero\_id, self.df[f'radiant\_player\_{1}\_hero\_id']], ignore\_index=True).drop\_duplicates()

self.df\_hero\_id = pd.concat([self.df\_hero\_id, self.df[f'dire\_player\_{1}\_hero\_id']], ignore\_index=True).drop\_duplicates()

mas\_del = ['league', 'league\_id', 'league\_start\_date\_time', 'league\_end\_date\_time', 'league\_region',

'series\_id', 'series\_type', 'match\_id',

'match\_start\_date\_time', 'radiant\_team\_name', 'dire\_team\_name',

'radiant\_player\_1\_hero', 'radiant\_player\_1\_position', 'radiant\_player\_1\_lane',

'radiant\_player\_1\_role', 'radiant\_player\_1\_name',

'radiant\_player\_2\_hero', 'radiant\_player\_2\_position', 'radiant\_player\_2\_lane',

'radiant\_player\_2\_role', 'radiant\_player\_2\_name',

'radiant\_player\_3\_hero', 'radiant\_player\_3\_position', 'radiant\_player\_3\_lane',

'radiant\_player\_3\_role', 'radiant\_player\_3\_name',

'radiant\_player\_4\_hero', 'radiant\_player\_4\_position', 'radiant\_player\_4\_lane',

'radiant\_player\_4\_role', 'radiant\_player\_4\_name',

'radiant\_player\_5\_hero', 'radiant\_player\_5\_position', 'radiant\_player\_5\_lane',

'radiant\_player\_5\_role', 'radiant\_player\_5\_name',

'dire\_player\_1\_hero', 'dire\_player\_1\_position', 'dire\_player\_1\_lane', 'dire\_player\_1\_role',

'dire\_player\_1\_name',

'dire\_player\_2\_hero', 'dire\_player\_2\_position', 'dire\_player\_2\_lane', 'dire\_player\_2\_role',

'dire\_player\_2\_name',

'dire\_player\_3\_hero', 'dire\_player\_3\_position', 'dire\_player\_3\_lane', 'dire\_player\_3\_role',

'dire\_player\_3\_name',

'dire\_player\_4\_hero', 'dire\_player\_4\_position', 'dire\_player\_4\_lane', 'dire\_player\_4\_role',

'dire\_player\_4\_name',

'dire\_player\_5\_hero', 'dire\_player\_5\_position', 'dire\_player\_5\_lane', 'dire\_player\_5\_role',

'dire\_player\_5\_name',

'game\_version\_id', 'league\_tier']

for i in mas\_del:

del self.df[i]

# Обучение модели

def train(self,n):

df = self.df.head(n)

# Заполнение пропусков

for column in df.columns:

if df[column].dtype == 'object': # Если это строка

df[column].fillna('Unknown', inplace=True)

elif pd.api.types.is\_numeric\_dtype(df[column]): # Если это числовой тип

df[column].fillna(0, inplace=True)

elif pd.api.types.is\_datetime64\_any\_dtype(df[column]): # Если это datetime

df[column].fillna(pd.NaT, inplace=True)

# Целевая переменная

X = df.drop(['winner\_id'], axis=1) # Все столбцы, кроме winner\_id

y = df['winner\_id'] # Целевая переменная

y = (X['radiant\_team\_id'] == df['winner\_id']).astype(int) + 1

# Разделение данных на обучающую и тестовую выборки

X\_train, self.X\_test, y\_train, self.y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Обучение модели

model = RandomForestClassifier(random\_state=42)

self.s = model.fit(X\_train, y\_train)

# Сохранение обучения модели

def save\_model(self):

dump(self.s, 'D:/model/model\_fit.joblib')

# Загрузка обученной модели

def load\_model(self):

self.s = load('D:/model/model\_fit.joblib')

# Результат

def result(self, data):

X = self.search\_id(data)

X = pd.DataFrame([X])

X = X.drop(['winner\_id'], axis=1)

pred = self.s.predict(X)

if pred == self.radiant\_team[1]:

return pred

else: return pred

def accuracy\_test\_data(self):

y\_pred = self.s.predict(self.X\_test)

accuracy = accuracy\_score(self.y\_test, y\_pred)

return accuracy

def search\_id(self, data):

mas\_team = [self.df\_team[0].to\_list()] + [self.df\_team\_id[0].to\_list()]

mas\_players = [self.df\_player\_name[0].to\_list()] + [self.df\_player\_id[0].to\_list()]

mas\_hero = [self.df\_hero[0].to\_list()] + [self.df\_hero\_id[0].to\_list()]

mas\_team[0] = [team for team in mas\_team[0] if pd.notna(team)]

mas\_players[0] = [team for team in mas\_team[0] if pd.notna(team)]

mas\_hero[0] = [team for team in mas\_team[0] if pd.notna(team)]

test = self.create\_slovar(self.df.columns.to\_list())

# Проверка на NaN для радиант команды

if pd.notna(data.get('radiant\_team\_name')) and data['radiant\_team\_name'] in mas\_team[0]:

test['radiant\_team\_id'] = mas\_team[1][mas\_team[0].index(data['radiant\_team\_name'])]

else:

test['radiant\_team\_id'] = 1 # или любое другое значение по умолчанию

# Проверка на NaN для дайр команды

if pd.notna(data.get('dire\_team\_name')) and data['dire\_team\_name'] in mas\_team[0]:

test['dire\_team\_id'] = mas\_team[1][mas\_team[0].index(data['dire\_team\_name'])]

else:

test['dire\_team\_id'] = 2 # или любое другое значение по умолчанию

self.dire\_team = [data.get('dire\_team\_name', 'Unknown'), test.get('dire\_team\_id', None)]

self.radiant\_team = [data.get('radiant\_team\_name', 'Unknown'), test.get('radiant\_team\_id', None)]

for i in range(1, 6):

# Проверка на NaN для радиант игроков

radiant\_player\_name = data.get(f'radiant\_player\_{i}\_name')

radiant\_player\_hero = data.get(f'radiant\_player\_{i}\_hero')

if pd.notna(radiant\_player\_name) and radiant\_player\_name in mas\_players[0]:

test[f'radiant\_player\_{i}\_id'] = mas\_players[1][mas\_players[0].index(radiant\_player\_name)]

else:

test[f'radiant\_player\_{i}\_id'] = None

if pd.notna(radiant\_player\_hero) and radiant\_player\_hero in mas\_hero[0]:

test[f'radiant\_player\_{i}\_hero\_id'] = mas\_hero[1][mas\_hero[0].index(radiant\_player\_hero)]

else:

test[f'radiant\_player\_{i}\_hero\_id'] = None

# Проверка на NaN для дир игроков

dire\_player\_name = data.get(f'dire\_player\_{i}\_name')

dire\_player\_hero = data.get(f'dire\_player\_{i}\_hero')

if pd.notna(dire\_player\_name) and dire\_player\_name in mas\_players[0]:

test[f'dire\_player\_{i}\_id'] = mas\_players[1][mas\_players[0].index(dire\_player\_name)]

else:

test[f'dire\_player\_{i}\_id'] = None # или любое другое значение по умолчанию

if pd.notna(dire\_player\_hero) and dire\_player\_hero in mas\_hero[0]:

test[f'dire\_player\_{i}\_hero\_id'] = mas\_hero[1][mas\_hero[0].index(dire\_player\_hero)]

else:

test[f'dire\_player\_{i}\_hero\_id'] = None

return test

def create\_slovar(self,headers):

dictionary = {header: np.nan for header in headers}

return dictionary

bot.py

import telebot  
from telebot import types  
import dota\_ai  
import spacy  
import text\_parser  
  
nlp = spacy.load("ru\_core\_news\_sm")  
  
ai = dota\_ai.dota\_ai()  
n = 200  
ai.train(n)  
mas\_hero = ai.df\_hero[0].to\_list()  
user\_data = {}  
s = '\n'.join(mas\_hero[:-1])  
  
bot = telebot.TeleBot('7692907060:AAGSg9M2y1kqfJBCa1TeuWBzF1mbTwrx-Qg')  
  
@bot.message\_handler(commands=['start'])  
def start(message):  
 markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)  
 btn1 = types.KeyboardButton("Начать использование")  
 btn2 = types.KeyboardButton("Помощь")  
 markup.add(btn1, btn2)  
 bot.send\_message(message.from\_user.id, "Данный бот может использоваться для предсказания матчей по Dota2", reply\_markup=markup)  
  
@bot.message\_handler(content\_types=['text'])  
def handle\_text(message):  
 *"""Обрабатывает текстовые сообщения от пользователя."""* user\_input = message.text.lower()  
 doc = nlp(user\_input)  
  
 # Распознавание намерений пользователя  
 if any(token.lemma\_ == "помощь" for token in doc):  
 show\_help(message)  
 elif any(token.lemma\_ == "начать" for token in doc):  
 start\_usage(message)  
 elif any(token.lemma\_ == "точность" for token in doc):  
 bot.send\_message(message.from\_user.id, f"Точность модели: {ai.accuracy\_test\_data()}")  
 elif any(token.lemma\_ == "герой" for token in doc):  
 bot.send\_message(message.from\_user.id, s)  
 elif any(token.lemma\_ in ["предсказать", "результат"] for token in doc):  
 bot.send\_message(message.from\_user.id, "Введите данные для предсказания результата матча.")  
 user\_data[message.from\_user.id] = 'waiting\_for\_prediction\_data'  
 elif user\_data.get(message.from\_user.id) == 'waiting\_for\_prediction\_data':  
 user\_input = message.text  
 try:  
 parsed\_data = text\_parser.parse\_input\_text\_to\_data(user\_input)  
 bot.send\_message(message.from\_user.id, f"Данные успешно обработаны")  
 prediction = ai.result(parsed\_data)  
 winner = "Radiant" if prediction == 1 else "Dire"  
 bot.send\_message(message.from\_user.id, f"Предсказание: победит команда {winner}.")  
 video\_path = 'Но это не точно.mp4'  
 bot.send\_video(message.from\_user.id, open(video\_path, 'rb'))  
 except Exception as e:  
 bot.send\_message(message.from\_user.id, "Ошибка при обработке данных. Проверьте формат ввода.")  
 print(f"Ошибка: {e}")  
 else:  
 bot.send\_message(message.from\_user.id, "Извините, я не понял ваш запрос. Попробуйте выбрать действие из меню.")  
  
def show\_help(message):  
 *"""Отображает список функций и инструкций по использованию бота."""* help\_text = (  
 "⚙️ \*\*Список доступных команд и функций бота\*\*:\n\n"  
 "1️⃣ \*\*Вывести точность\*\*\n"  
 " - Узнать текущую точность модели предсказания.\n\n"  
 "2️⃣ \*\*Предсказать результат матча\*\*\n"  
 "3️⃣ \*\*Вывести список героев\*\*\n"  
 " - Просмотр всех доступных героев в системе.\n\n"  
 " - Введите данные о командах, игроках и героях для прогноза.\n"  
 " - Формат ввода будет показан перед использованием.\n\n"  
 " - Пример входных данных: Команда Radiant: Invincible Warriors. Игроки: Player1 на Luna, Player2 на Rubick, Player3 на Beastmaster, Player4 на Sniper, Player5 на Lich. Команда Dire: Shadow Fiends. Игроки: PlayerA на Mirana, PlayerB на Wraith King, PlayerC на Dark Willow, PlayerD на Ogre Magi, PlayerE на Zeus."  
 "❓ Если у вас возникли вопросы, начните ввод текста, и бот попытается понять ваш запрос."  
 )  
 bot.send\_message(message.from\_user.id, help\_text, parse\_mode='Markdown')  
  
def start\_usage(message):  
 markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)  
 btn1 = types.KeyboardButton('Вывести точность')  
 btn2 = types.KeyboardButton('Предсказать результат матча')  
 btn3 = types.KeyboardButton('Вывести список героев')  
 markup.add(btn1, btn2, btn3)  
 bot.send\_message(  
 message.from\_user.id,  
 "Выберите одно из доступных действий:",  
 reply\_markup=markup,  
 )  
  
  
bot.polling(none\_stop=True, interval=0)

text\_parser.py

import spacy  
  
# Загрузка модели spaCy  
nlp = spacy.load("ru\_core\_news\_sm")  
  
def parse\_input\_text\_to\_data(text):  
  
 data = {  
 "radiant\_team\_name": None,  
 "dire\_team\_name": None,  
 "radiant\_player\_1\_hero": None,  
 "radiant\_player\_1\_name": None,  
 "radiant\_player\_2\_hero": None,  
 "radiant\_player\_2\_name": None,  
 "radiant\_player\_3\_hero": None,  
 "radiant\_player\_3\_name": None,  
 "radiant\_player\_4\_hero": None,  
 "radiant\_player\_4\_name": None,  
 "radiant\_player\_5\_hero": None,  
 "radiant\_player\_5\_name": None,  
 "dire\_player\_1\_hero": None,  
 "dire\_player\_1\_name": None,  
 "dire\_player\_2\_hero": None,  
 "dire\_player\_2\_name": None,  
 "dire\_player\_3\_hero": None,  
 "dire\_player\_3\_name": None,  
 "dire\_player\_4\_hero": None,  
 "dire\_player\_4\_name": None,  
 "dire\_player\_5\_hero": None,  
 "dire\_player\_5\_name": None  
 }  
  
 # Обрабатываем текст через spaCy  
 doc = nlp(text)  
  
 radiant\_counter = 1  
 dire\_counter = 1  
 current\_team = None  
  
 for sent in doc.sents:  
 if "Команда Radiant" in sent.text:  
 # Извлекаем название команды Radiant  
 data["radiant\_team\_name"] = sent.text.split(":")[1].strip().rstrip('.')  
 current\_team = "Radiant"  
  
 elif "Команда Dire" in sent.text:  
 # Извлекаем название команды Dire  
 data["dire\_team\_name"] = sent.text.split(":")[1].strip().rstrip('.')  
 current\_team = "Dire"  
  
 # Проверяем наличие "Игроки:" в предложении  
 if "Игроки:" in sent.text:  
 players\_and\_heroes = sent.text.split("Игроки:")[1].strip().rstrip('.').split(", ")  
 for entry in players\_and\_heroes:  
 if " на " in entry:  
 player, hero = entry.split(" на ")  
 if current\_team == "Radiant":  
 data[f"radiant\_player\_{radiant\_counter}\_hero"] = hero.strip()  
 data[f"radiant\_player\_{radiant\_counter}\_name"] = player.strip()  
 radiant\_counter += 1  
 elif current\_team == "Dire":  
 data[f"dire\_player\_{dire\_counter}\_hero"] = hero.strip()  
 data[f"dire\_player\_{dire\_counter}\_name"] = player.strip()  
 dire\_counter += 1  
  
 return data